KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

# Large-scale Classification and Retrieval of 3D Shapes

**Jan Knopp**

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

May 2015

# Large-scale Classification and Retrieval of 3D Shapes

**Jan KNOPP**

Supervisory Committee:
Prof. dr. ir. Jean Berlamont, chair
Prof. dr. ir. Luc Van Gool, supervisor
Prof. dr. ir. Tinne Tuytelaars
Prof. dr. ir. Philip Dutré
Prof. dr. ir. Theo Moons
Prof. dr. ir. Remco C. Veltkamp
(Universiteit Utrecht)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

May 2015

Uitgegeven in eigen beheer, Jan Knopp, Kasteelpark Arenberg 10 box 2441, B-3001 Heverlee (Belgium)

*To my grandma*

# Preface

I would like to thank my supervisor, Prof. Luc Van Gool, for opportunity to work in VISICS group. I have learned many things about how to conduct research, write papers and a great deal more.

I would also like to thank Mukta Prasad for our in-depth discussions about almost every subject that I touched on my doctoral studies, cheering me up, and being a kind virtual mother to me.

I am also very thankful to members of the examination committee, Prof. Jean Berlamont, Prof. Tinne Tuytelaars, Prof. Philip Dutré, Prof. Theo Moons and Prof. Remco C. Veltkamp. Their feedback helped to improve the quality of this thesis greatly.

I would like to express my deep gratitude to Prof. Thomas Funkhouser and Sid Chaudhuri, who have shown me the world of Computer Graphics, taught me a lot of about research, and encouraged collaboration that was extremely enjoyable, interesting and open minded. Also I am grateful for hosting me like a member of their own family.

I am also very thankful to Tomas Pajdla and Josef Sivic for introducing me to the field of Computer Vision. I would not have been able to do everything that I have done without the background I have gained under their supervision.

My deepest thanks also go to all the staff of the VISICS group for both, being professional colleagues during the research time, and for being great friends. It would have been truly impossible to have had such great years of PhD study without them. Especially, I would like to thank the colleague of September '13 and the doctor of November '13 for sharing the lab during the endless deadline nights and having a lot of fun.

My thank also have to go to all friends in Leuven for being great friends, open-minding law discussions, skating, delicious Thai food, Seven Oaks beers, sushi, allowing me to stay at the best place ever, biking, playing guitar, squash, ice hockey, and running too.

I would like to thank my Czech friends for being the same and still inviting me with open hands.

Finally, I owe an indescribable amount of gratitude to my parents, all of my brothers, my nieces, my cousins, my aunt, my uncle.

# Abstract

This thesis focuses on three main topics in the area of 3D recognition: shape classification, retrieval, and scene segmentation.

For classification, in contrast to previous methods that focused on the overall shape appearance only, we extended the popular SURF features into a third dimension and presented a novel method to learn the efficient 3D Implicit Shape Model (3D ISM), which is a class-specific star model that combines the appearance and the relative position of local patches from 3D shapes.

The proposed shape retrieval approach is based on our previous classification model. Thus, 3D ISM introduces additional spatial constraints to improve the basic bag-of-visual-words ordering. This ISM-based verification step allows us to use shape expansion that gains even better results. In addition, we show how to use this retrieval pipeline to improve shape matching, classification, and text-based 3D shape search. The method was also practically used in the 3D Coform project in a search for artifacts in a museum's repository.

The previously introduced method for classification was only applied to recognize classes of shapes that are isolated, clean, and without holes. Instead, segmenting large 3D scenes is a more realistic scenario. Objects need to be firstly found and then segmented. We propose two methods. The first, combines our previous findings in classification with CRF optimization and the second is a more sophisticated framework that solves ISM and graph optimization jointly. Once the object is found and segmented from its background, we use our method (3D ISM) or introduced new Boltzmann Machines-based approach to fill-in holes and to correct the wrong parts of incomplete shape.

Each task is evaluated on several benchmarks against a variety of state-of-the-art methods.

# Beknopte samenvatting

Dit proefschrift richt zich op drie belangrijke onderwerpen op het gebied van 3D herkenning: vorm classificatie, opvraging en de segmentatie van scénes.

Voor classificatie breidden we, in tegenstelling tot bestaande methoden die zich enkel richten op het uitzicht van de totale vorm, de populaire SURF features uit naar een derde dimensie en stelden we een nieuwe methode voor om het efficiente 3D Impliciete Vorm Model (3DISM) te leren. Dit is een klasse-specifiek ster model dat het uitzicht en de relatieve positie van locale beeldstukjes (patches) van 3D vormen combineert. De voorgestelde aanpak voor vorm opvraging is gebaseerd op ons eerdere classificatie model.

3DISM introduceert op deze manier extra ruimtelijke beperkingen om de volgorde van de standaard bag-of-visual-words te verbeteren. Deze ISM gebaseerde verificatie stap stelt ons in staat om vorm expansie te gebruiken die nog betere resultaten krijgt. Daarnaast laten we zien hoe deze opvragingspijplijn te gebruiken om vorm matching, classificatie, en tekstgebaseerde 3D-vorm opzoeking te verbeteren. De werkwijze is ook in de praktijk gebruikt in het 3D Coform project in een zoektocht naar artefacten in de opslagplaats van een museum.

De eerder geïntroduceerde methode voor classificatie was enkel toegepast om klassen te herkennen van mooie geisoleerde vormen zonder gaten. Het segmenteren van grote 3D scénes is echter een meer realistisch scenario. Objecten moeten eerst gevonden worden en dan gesegmenteerd. We stellen twee methoden voor. De eerste combineert onze eerdere bevindingen in classificatie met CRF optimalisatie en de tweede is een meer geavanceerd framework dat gezamenlijk ISM en grafiek optimalisatie oplost. Zodra het object wordt gevonden en gesegmenteerd is van de achtergrond, gebruiken we onze methode (3DISM) of de geintroduceerde nieuwe aanpak gebaseerd op Boltzmann Machines om gaten in te vullen en om de verkeerde delen van de incomplete vorm te corrigeren.

Elke taak wordt geëvalueerd op verscheidene benchmarks tegen een variëteit van state-of-the-art methoden.

# Glossary

| Notation | Description |
|---|---|
| $\mathbf{a}$ | vectors in lowercase latters |
| $\mathsf{A}$ | matrix in sans serif uppercase latters |
| $\mathsf{A}^{-1}$ | inverse of matrix $\mathsf{A}$ |
| $\mathsf{A}^{\top}$ | transpose of matrix $\mathsf{A}$ |
| $\mathcal{A}$ | sets in uppercase calligraphic letters |
| $\lvert\mathcal{A}\rvert$ | number of elements in the set $\mathcal{A}$ |
| $p()$ | probability |
| $\ell \in \mathcal{L}$ | class label the set of labels $\mathcal{L}$ |
| $\mathbf{i} \in \mathcal{I}$ | feature $\mathbf{i}$ from the set of features $\mathcal{I}$ |
| $\mathbf{i} = \{\mathbf{p}_i, \sigma_i, \mathbf{d}_i, c_i\}$ | feature/vertex/voxel/3D point $\mathbf{i}$, it is usually a tuple of position ($[x_i, y_i, z_i]^{\mathsf{T}}$), scale, descriptor, and quantized descriptor (visual word) |
| $x_i \in \mathcal{L}$ | class label of the point $\mathbf{i}$ from the set of labels $\mathcal{L}$ |
| $c_i \in \mathcal{C}$ | cluster center (visual word) from the set of all cluster centers of size $\lvert\mathcal{C}\rvert$ |
| $\mathbf{i}^* = \{\mathbf{p}_i^*, \sigma_i^*, \mathbf{d}_i^*, c_i^*\}$ | feature/vertex/voxel/3D point $\mathbf{i}^*$ from the training set |
| $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ | vote cast from the set of vote casts |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Object detection and recognition is considered one of the ultimate goals of artificial intelligence. Despite preliminary studies indicating the advantages of 3D object representation [126, 22, 108, 138], especially in the detection of object boundaries and segmentation [126], the last decades of computer vision research mostly focused on pure 2D image signals (i.e. [171, 49, 92, 141]). In this area of analyzing 2D data, object classification, detection, segmentation, and retrieval have reached great results, but are still far from outstanding [92]. Unfortunately, they remain dependent upon the pure image data.

Fueled by years of successful usage in the movie industry, the recent rise of 3D capturing systems [72], easy-to-use Structure from Motion algorithms [186], and the availability of large 3D repositories [62], the need for 3D data understanding remains an area of interest. It is only now that state-of-the art algorithms benefit from the 3D data. Unfortunately, most methods take the advantage of only a pixel's position in 3D [140, 139, 106, 159], or the 3D geometry that is around pixels [12]. Thus, considering all information that can 3D give for recognition is still rare [174, 61]. The thesis focuses on this 3D understanding.

In this Chapter, we first give motivation for focusing on analyzing 3D data in §1.1. Then, in §1.2, we discuss the challenges. Finally, we conclude the Chapter with the thesis overview in §1.3, where the structure and contributions of the thesis are described.

## 1.1 Motivation and objectives

The amount of available visual data has been continuously growing in the last years. For example, Facebook users upload about 300M images per day [60]. Another example is the biggest benchmark for image classification, ImageNet [40], which contains millions of labeled pictures. The amount of 3D data is also significantly growing. Google stores images with 3D information of almost the entire world (its subset is available via popular Google StreetView [64] and Google Earth [63]); museums constantly increase their digitization power; Kinect [72] allows anyone to collect 3D data of their surroundings; etc. These databases store more information than one can ever see in a lifetime. The amount of the data makes these datasets extremely interesting, while processing them and retrieving useful information becomes more of a challenge than ever before.

Unsupervised information discovery and 3D learning opens doors for several useful applications. It motivates the development of algorithms that understand 3D data from the aforementioned datasets. For instance, robots, or autonomous cars can understand the environment; surrounding objects can be automatically detected and their purpose and properties can be retrieved from the database. These objects of interest can be segmented from their background, taken out of the scene, cleaned, and put back to the scene, so the scene will be without missing parts and more suitable for e.g. virtual reality. In another example, museums now digitize their archives. Artifacts can then be compared to the rest of the archive with the goal of finding similar objects or retrieving new information. Thanks to the 3D representation of the data, repository processing can be accomplished remotely without any need for the person's presence in the museum. In computer graphics, there is a huge effort for automatic animating 3D modeling, or realistic scene modeling and rendering that are like a real photograph. However, these processes are time consuming, still not perfect, and require a human in the loop [77]. Again, learning useful data from the datasets can help significantly.

All these applications motivate our interest in understanding the content of 3D datasets. Our goals can be summarized in three points: i) automatic categorization of the previously unseen objects; ii) retrieval of a 3D query from a large database; iii) detecting, segmenting and gleaning objects in 3D scenes.

Methods for previously discussed applications need a meaningful description of the 3D objects based on the semantic and context. A need for generalization over variations and noise becomes necessary too. More specifically, one would expect that the pure appearance of object parts (e.g. a person has a head) will not be enough, while their relative positions (e.g. head is above the torso)

cat lion lion

Figure 1.1: **Similarity between objects.** The same pose of the different object class makes them globally similar (first two shapes: cat vs. lion), while the appearance of the local parts, head or paw, emphasizes higher similarity between lions (last two shapes).

should play an important role. A precise model of the part relations should be learned automatically where it should be considered that some parts are more important than others. Thus, we learn such a model and use it in several applications to fulfill the goals of the thesis.

## 1.2 Why is 3D recognition challenging?

The previously discussed goals of this thesis are difficult problems that even humans are not always successful in solving. In this section, we will review the challenges this thesis deals with and we will link them to the relevant parts of the thesis.

**Representation of 3D objects.** No matter whether we work with data of text, images, sounds, or 3D shapes, the representation of the data we work with is crucial. To be reliable, the representation needs to be invariant to certain variations and noise of the data that may appear. For example, the representation of an image for object detection should be invariant to variables such as camera view point and image resolution [108, 14]. In 3D, we would like to get the same results independent of the level of detail of a 3D object, its rotation, or even if some data are missing. This is highly relevant to understanding whether we should focus on local or global parts, and which objects are relevant to each other. For example, while a cat differs from a lion, different poses of objects may cause similarity between different classes, while this may increase differences within the class (fig. 1.1). Data representation often defines the cornerstone of the whole pipeline as further methods depend on the representation. We address shape representation in several parts of the thesis. In §3.2, we propose a shape description that is robust to missing data. In §3.3, we describe a design of our 3D ISM class model to measure global dependencies of local parts, and we show how to estimate the potentially missing (or incorrect) parts in Chapter 8.

**Real life data.** Several methods perform almost perfectly on benchmarks for 3D search, recognition, or segmentation. Unfortunately, these methods [129, 29,

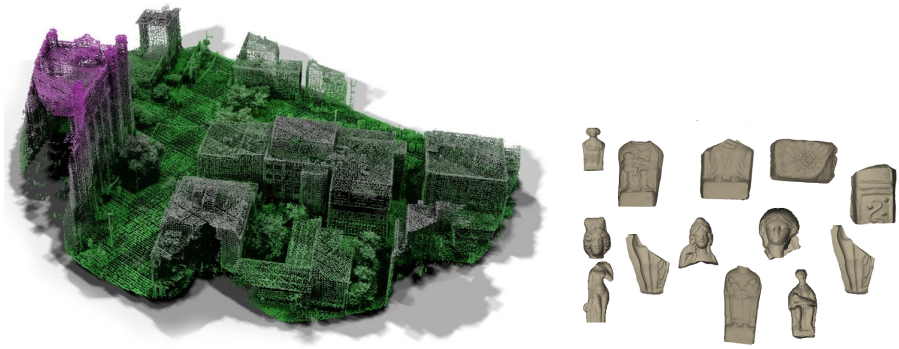


Figure 1.2: **Example of large data that we process.** Left: a part of the scan of Ottawa city [61]. Color represents height of the point. Right: several 3D shapes from the set of the sculptures from a museum [4].

131, 76] often assume clean and manually hand-crafted 3D shapes. In contrast, incomplete shapes, wrong position of some vertices, missing data, occlusions, background etc. often appear in realistic scenarios. In §3.2, a method for representing shapes that is robust to missing points is introduced. Our model that learns class-specific properties is then evaluated on a real-life noisy dataset in Chapter 7.

**Scalability.** Real-life applications always introduce strong requirements for the speed and/or memory of the algorithm. In analyzing 3D, it is a case of processing large city-scale scans, searching in vast databases of shapes, or processing shapes with high level of detail (fig. 1.2). Algorithms have two parts with different requirements. While the training (offline) part of the algorithm does not have strong requirements, it is still convenient to store data in an efficient way that allows fast access and to propose algorithms that can be trained rapidly. The test part has a significantly stronger requirement as the result needs to be given as soon as possible. Unfortunately, there is a natural trade-off between speed and accuracy that must be always taken into consideration. We deal with large-scale problems on the task of shape retrieval in Chapter 3 and in Chapter 5, and on the task of shape detection/segmentation in Chapter 7.

## 1.3 Contributions & Overview

This thesis proposes algorithms that focus on tackling the challenges mentioned above. Specifically, we focus on the following two:

(i) Given a new unseen 3D shape, we estimate its class (this task is called classification or class recognition) and we find relevant shapes in the database (retrieval).

(ii) Given a 3D scene, we focus on detecting the visible objects, segmenting them from the background, filling in parts that are missing, and improving those parts that are noisy.

For the first application, we extend the popular 2D SURF [14] feature detector/descriptor to 3D and take advantage of representing objects by a set of local features associated with the geometry and relative position to the center. Using this, we show state-of-the-art results in classification and we also introduce new constraints that help in retrieval. For the second application, we used our previous findings and combined them for joint detection and segmentation. Completion was achieved by investigating the power of deep learning (especially Restricted Boltzmann Machines) in 3D. Relation between parts of this thesis is in fig. 1.3. We now summarize the contributions and link them to the corresponding chapters.

**Shape description and classification (Chapters 3 and 4).** We first introduce the extension of the popular 2D SURF into 3D for robust 3D shape feature detection and description. Then, we show how to use spatial constraints for representing the class of the 3D shapes and how to use them for shape classification. This results in the 3D Implicit Shape Model (3D ISM) that was presented in [85]. Community's re-implementation of 3D ISM is included in the popular open point cloud library [154]. The work is described and tested in Chapter 3. In addition to this, we have improved our classification method by rotation invariant voting, which was presented in [86] and is summarized in Chapter 4.

**Shape retrieval (Chapter 5).** We combined previous findings on shape description using 3D SURF and shape classification using 3D ISM for the task of shape retrieval. This work was published in [84]. Proposed methods were also used to improve shape classification with limited training data, shape matching or text-based shape search. The method was used practically in the 3D Coform shape search tool [4]. This project focused on 3D analysis of cultural heritage data. Our method contributed in the area of searching in datasets of 3D shapes of sculptures.

**Scene segmentation (Chapters 6 and 7).** Previously described 3D ISM was only used to recognize the class of an isolated 3D shape. Analyzing complete

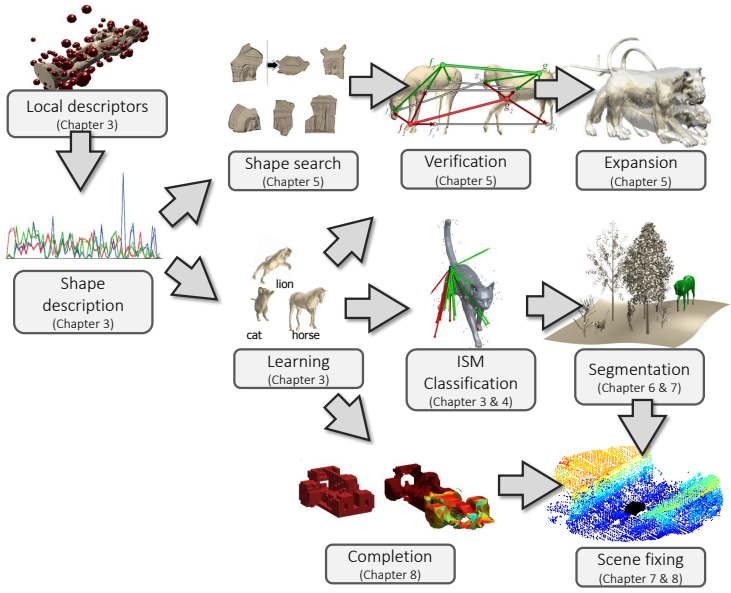



Figure 1.3: **Dependencies between parts of the dissertation.** We introduce a method to detect and describe features of the 3D shape. The shape representation as a set of local features is used to learn a spatial model of the class of objects for classification, or for retrieval in large databases. Such classifications are also applied to improve retrieval, to detect and segment objects in 3D scenes, as well as to complete partial shapes. For each part, we discuss state-of-the-art methods and compare them with our novel proposals on a variety of benchmarks.

3D scenes with the goal of finding objects and segmenting them from the background is a more realistic and challenging application. A combination of 3D ISM and the graph optimization technique for scene segmentation and object detection is presented in Chapter 6 and was published in [83]. Then, we also show a more appropriate solution that uses the inference between the currently detected objects and points labels. The method is described in Chapter 7.

**Scene completion (Chapter 8).** Previous methods detect objects in realistic scenes and segment these objects from the background. If the 3D scene represents a realistic world, it often suffers from noise and missing data. We use our method to improve these incomplete 3D objects, but one would need a more robust model than 3D ISM to generalize and to guess the structure of a missing part of the shape. To achieve this, we explored Restricted Boltzmann Machines for completing partial shapes against 3D ISM and other competitors.

# Chapter 2

# Background

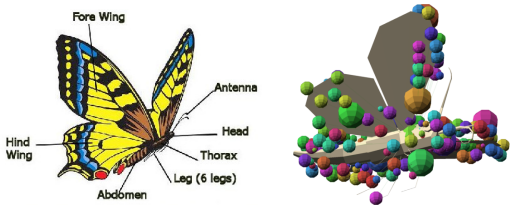
Due to the large interconnection of our work with the numerous findings in computer vision, graphics and machine learning, this chapter separately introduces each problem the thesis deals with and we discuss only the intuitive (naive) baselines for each problem. The introduction of the naive baselines is very useful to highlight the challenges of each task and discuss the evaluation methods. Detailed descriptions of state-of-the-art methods are in the corresponding chapters of the thesis, rather than in this chapter as it is used to introduce details of the problem, notation and our proposal.

The chapter starts with discussing the representation of 3D data in §2.1.1. In §, we discuss early methods for 3D recognition. Then, we have an overview of methods to capture real-life objects in 3D in §2.1.3, and we continue with approaches to describe 3D shapes in §2.1.4. The scheme of the problem introduction and its description driven by discussing the intuitive baseline is applied for retrieval in §2.2, classification in §2.3, and segmentation in §2.4. The chapter is concluded with §2.5 which describes 3D datasets.

## 2.1 Representing real-life objects as 3D shapes

### 2.1.1 What are objects and their categories?

Early studies about the object classes and the relevance between objects [151, 3] are highly relevant to problems such as how to learn the model for a specific category of objects, which properties this model should have, what is the




(a) (b)

Figure 2.1: **Example of butterfly's forms.** (a) Shows human marked parts that describe the object. This is related to the object's forms [151]. (b) 3D shape with colored spheres that corresponds to the visual words [85]. Courtesy of [1].

similarity between shapes, and how to find similar or relevant models from the dataset. In the context of this thesis, we would like to compare the exemplar theory [111, 116] with the theory of forms [151, 3]. To see their relation, let us assume an example of how to represent a set of butterflies. In the exemplar theory, the butterfly class is simply a set of all previously seen butterflies, while in the theory of forms we need to extract a butterfly's specific ideals/forms such as its color, antennas on the head, number of legs etc. Some of our methods follow the approach of the theory of forms. Further recognition of the previously unseen object depends on the object representation. The theory of forms explains a new object by the set of forms of the ideal butterfly (fig. 2.1(a)) such as: Does it have two wings? Are they colorful? While in the exemplar theory, the unknown object is compared with all instances of the previously seen butterflies and if we think that the unknown object is similar to one of those previously seen, it will be the butterfly.

The thesis extends the work of Bellard *et al.* [9] and Leibe *et al.* [104] into 3D Implicit Shape Model (3D ISM). In the off-line part, we learn forms (visual words in our case) of a class and their relation to the object, i.e. butterfly has antennas in the front (fig. 2.1(b)).

### 2.1.2 Former 3D recognition

There has a been a lot of research on how humans represent and recognize 3D objects and 3D scenes [113]. Pioneers of vision and graphics were already treating the image as the projection of the 3D objects [22, 108, 138]. Nevatia and Binford [126] found that 3D position of image pixels leads to significant improvement of detecting the boundaries of objects as well as segmenting it.

Early graphics and vision research went further in 3D to analyze whole scenes and relations between objects [16]. For example, Kuan and Drazovich [93] considered cylinders as the basic elements for a scene (fig 2.2). Then, a scene is divided into several levels: basic edges, then surfaces, cylinders, and finally relations between objects. This representation of the scene is

then applied for scene recognition. Gennery [59] represented scene objects by ellipsoids and he found this very useful to approximate complex scenes. Visionary work of Horaud and Bolles [69] suggested to focus on the local parts as CAD data are often very similar and local patches would hold the promise to differentiate between objects. Even more, they suggested to use only a subset of the most discriminate local parts and their pipeline consisting of feature detection, clustering and hypothesis generation.

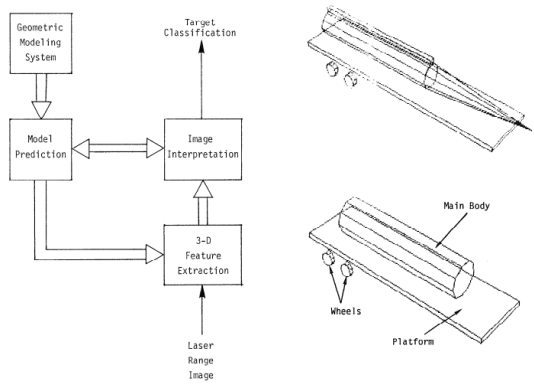



Figure 2.2: **3D recognition system of Kuan and Drazovich [93].** Left: the system. Right-top: component level model. Right-bottom: decoy.

Another interesting approach of Oshima and Shirai [130] presented a 3D recognition pipeline that groups points into small surfaces elements, classify elements either as planar or curved, merge them into objects and analyze the scene based on the relations between these objects.

### 2.1.3 3D to represent the real world

Processing 3D data has been already found very useful in solving several real-life problems. In this case, real-life means that we work with existing (real) objects. We here list examples of such problems and the real objects that are associated with them. *Retrieval* focuses on how these sculptures from the museum are relevant to the sculpture that was just found in the desert. *Classification* task aims at the recognition that the sculpture has a woman from the 16th century. *Segmentation* recognizes that this part of the sculpture is a vase. In this context, 3D data always corresponds to the real objects or a scene.

Thus, 3D objects are representations of the real world and the representability is crucial. There are two mainstream approaches to obtain digital 3D models. 3D object can be manually created in the 3D modeling software, or one can try to use specific software/hardware to do the reconstruction semi-automatically. In tab. 2.1, we show their advantages and disadvantages. We now discuss these methods in detail.

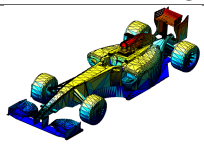
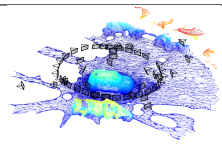
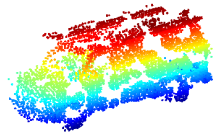
| | Manual crafting | SfM | LIDAR |
|---|---|---|---|
| Output |  |  |  |
| diff. to real | high | medium | small |
| quality | high | medium | medium |
| has holes | no | yes | yes |
| output | mesh | mesh | point cloud |
| time cost | expensive | medium | medium |
| effort | needs expert | medium | medium |
| $ | pay expert+SW | camera+SW | LIDAR |

Table 2.1: **Methods to obtain 3D data.** Methods for creating 3D object are listed in the first row. The table summarizes important properties for each method. Note that 3D shapes vary in a level of detail and noise.

**Hand-crafted shapes.** Manual creation of a 3D model according to the previously seen object is time consuming, needs a lot of expertise, suffers in reproducing details perfectly and, needs modeling software. Despite these disadvantages the manual crafting methods were the leading approaches to model 3D due to their popularity in the movie and gaming industry which introduced a huge effort to make this manual process as fast, and easy, as possible [29]. These hand-crafted 3D shapes are noiseless and without holes. They are crafted to look perfect for movies, but they do not have to be perfect copies of real objects. Until recently, hand-made models represented the largest 3D datasets. However, they are good for learning models (the databases are relatively large and clean). They are not appropriate when one wants to test methods for their robustness or to learn models that generalize over noise.

**Capturing real-life objects.** Instead of spending hours to create 3D objects that we want, one can go outside and directly digitize the surrounding world using specific hardware/software such as Kinect [72], LIDAR [61] or Structure-from-Motion algorithms [169, 186]. Thanks to these methods, datasets of real-life 3D scenes started to mature significantly too. The generated 3D models suffer from occlusion and noise, but they are extremely useful for navigation [180], world digitization [173], object recognition [31] etc. In summary, as these data are obtained more automatically, noise that makes them challenging to process always occurs, while the fact that they represent a camera's surrounding opens

up a broad variety of real-life applications. Thus, they are extremely interesting for research purposes.

### 2.1.4 Description of 3D shapes

Once we can obtain 3D data using the methods from the previous section §2.1.3, the data needs to be stored in the computer. Early research in 3D graphics suggested the following methods [16]:

(i) *Wireframe* where 3D objects are represented as points and edges between these points.

(ii) *Constructive solid geometry* where the storage data is a tree structure with nodes as basic primitives and branching nodes as boolean operands (intersection, union etc.).

(iii) *Spatial occupancy representation* in a voxel space or its smarter hierarchical modifications [26, 115] that allows for efficient data insertion, removal, or access operations.

(iv) *Surface boundary* which is a set of surfaces (sets of points, edges, faces) that defines the boundary of the object.

While these methods were proposed almost 40 years ago, today's computers store 3D data in the same way. The improvements focused on the algorithm or hardware (GPU) point of view in order to be more efficient in data processing or to significantly scale-up to be able to efficiently work with very large scenes [123], The basic ideas of these algorithms remain the same.

Any segmentation/retrieval/classification method requires a shape representation that can measure the similarity between two 3D shapes. This is inconvenient for the aforementioned representations. Thus, researchers would rather move away from pure 3D data storage formats to representations that are robust under certain transformations. Representation can be divided into main groups:

(i) *Global representation* is where some global information is extracted and the shape comparison can be performed by the alignment of these representations.

(ii) *Local representation* is where the shape is represented by the appearance of many local patches. Very often, results should be invariant to variations in scale, rotation, surface noise, and the pose of the shape.

For example, the 3D shape of the lion in fig. 1.1 should be closer to the lion in a different pose, rather than to the cat in the same pose. Additionally, the lion should be closer to the cat in the same pose than to the cat in a different pose. These requirements strongly support local description of shapes over global ones, as the cat and lion in the same pose look globally the same.

While local descriptors have numerous advantages, one needs to consider the fact that each shape will contain a different numbers of local descriptors which can make further processing more complicated and slower. The most popular method (and heavily used in this thesis) to overcome this problem is called **Bag of visual words (BoW)** [171, 21, 85, 141, 73, 181, 104] and we shortly describe it here. BoW uses descriptor clustering where $N$-dimensional local feature descriptors are projected into a one dimensional space. The method is implemented as follows: first, local feature descriptors are clustered into a set of distinct cluster centers $\mathcal{C}$, called a visual vocabulary. Each single feature is then associated with the visual word index of its closest cluster center. A shape is then encoded as a histogram of the number of times each vocabulary element shows up in it. Thus, each shape is represented by a sparse $|\mathcal{C}|$-dimensional vector. In the rest of this section, we will discuss specific descriptor types and how they have evolved over time.

In the early days of 3D graphics, the most popular method used a 3D object to define a 3D curve that is curved/sweeped by a cone and the intersection between the shape and the cone is defined in a specific way that measures the object's properties [176, 161]. Other methods that were considered as the most robust in the beginning of the twenty first century [53, 38] are based on renderings. Different views of the 3D object could be related via aspects graphs [88] or linear transformations [27]. Other pioneers of computer graphics focused on representing 3D data as skeleton graphs [183]. O'Rourke and Badler [128] proposed a method to described a shape based on how many (and which) spheres overlap the shape.

Global shape descriptions (e.g. based on global features) defined the state-of-the-art for a long time [179, 129, 53, 87]. Except for the prior work of Johnson and Hebert [74], only recently have local features started to attracted attention [85, 177, 21, 76, 170]. Some of these descriptors are visualized in fig. 2.3. The section continues with the overview of methods that we will later refer to, or that we will use as competitors against our proposals.

**Heat Kernel Signatures (HKS).** This geometry-curvature based method [177] uses a heat kernel to capture multi-scale isometric invariant geodesic information between features on a shape by characterizing it as the amount of heat transferred on the surface. This local descriptor has been shown to be stable under shape

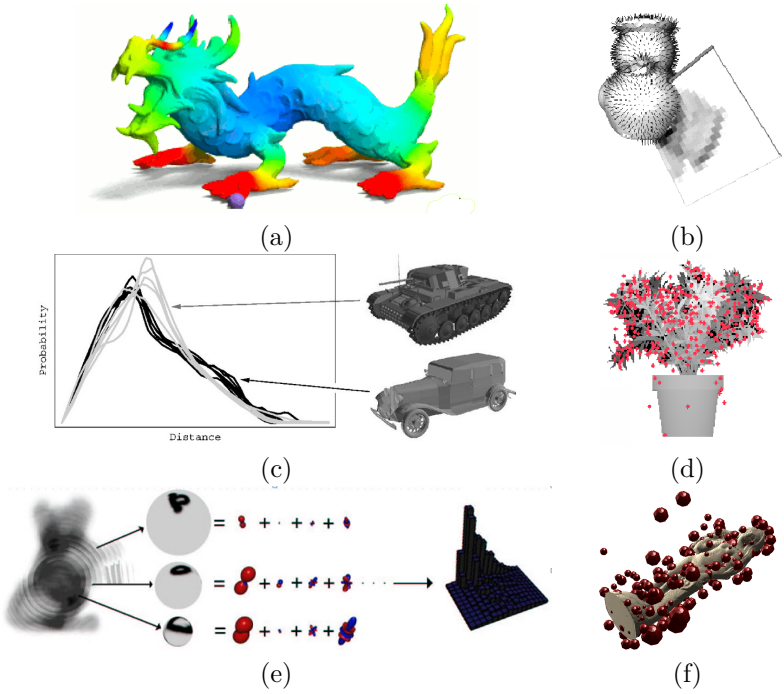



Figure 2.3: **Examples of 3D descriptors.** (a) HKS calculates the heat distribution on the shape (courtesy of Sun *et al.* [177]), (b) SI is based on the the histogram of points that intersects with the plane rotating along the point's normal (courtesy of Johnson *et al.* [74]), (c) Shape distribution measures point-to-point distances (courtesy of Osada *et al.* [129]), (d) 2.5D descriptors extract image feature at the rendered projections of 3D shapes (courtesy of Furuya *et al.* [53]), (e) SH calculates basis functions on the shape (courtesy of Kobbelt *et al.* [87]), (f) our 3D SURF calculates derivatives at the local neighborhood of the salient point.

perturbations from deformations. However, it suffers from noise and missing data.

**Spin Images (SI).** SI is a local descriptor [74] that records a spatial histogram of the 3D model's spatial occupancy around a 3D point. The histogram is aligned by surface orientation. We used a spin image size of 20% of the longest point-to-point distance on the shape and a spin-image resolution of $8 \times 8$, which results in a descriptor of 64 bins.

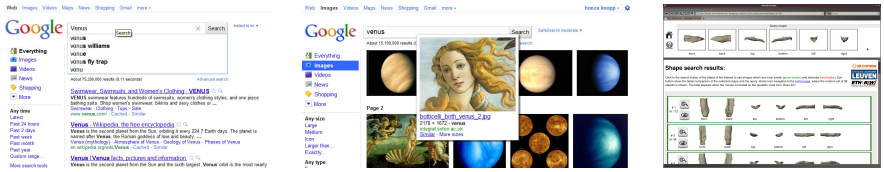


Figure 2.4: **Types of queries for the retrieval system.** An object (Venus in this case) can be found by different queries: Text query as in popular Google text search (left), Image query when we search for similar image (middle), or by a given 3D shape query (right).

**Shape Distribution (ShapeD2).** This global descriptor measures pairwise point to point distance distributions on the shape [129]. We used the implementation from the fvs library [2].

**2.5D SURF.** We also implemented a variation of the rendering-based methods [53, 38] that use 2D projections of 3D shapes (this is where 2.5D name came from). We render images from cameras around the vertices of the polyhedral convex hull. Then, 2D SURF features [14] are collected across all images for the subsequent generation of BoW vectors describing the shape globally. Note that 2D SURF is 2D orientation invariant and the collection of features from various viewpoints leads to the 3D orientation invariance in practice. We experimentally found that the method gives similar results to Furuya *et al.* [53].

**Spherical Harmonics (SH).** We use the original implementation of the rotation invariant SH expressed as a combination of basis functions on a sphere proposed by Kobbelt *et al.* [87].

## 2.2 Retrieval

Retrieval is defined as the activity of obtaining relevant data from the database (corpus), given a query [112]. The results of retrieval are often given in the form of a database ordering system such that relevant query objects are presented before the irrelevant ones [171, 34, 141] (fig. 2.4).

**Baseline.** If a relevance function between a pair of objects is defined, then we can calculate the similarity (inverse distance) between the query object and

each database object using the relevance function. The similarity value of each database object defines the ordering of the database and the retrieval results.

**Challenges.** The baseline method has several issues that makes it impossible to use.

(i) How do we define the relevance (similarity) function? It is sometimes hard for humans to decide which objects should be similar to each other (fig. 1.1). In the case of 3D shapes, it is important to deal with local versus global properties [179], noise, etc.

(ii) Even if we can come up with the precise relevance function, the estimation of the similarity between the query and millions of objects in the database becomes intractable. In practice, we always want results as fast as possible. The usual way to overcome these issues is the following: the relevance function is simplified [141], the data is clustered [171], and a more precise relevance function is used only on the most promising subset [141] of results. Herein, one always needs to deal with a trade-off between performance and time. When attempting to improve the performance, the algorithm is more costly and vice versa [141].

(iii) We have assumed clean and nice shapes so far. But imprecise data with noise and missing parts is typical for more realistic problems. Noise can worsen the performance significantly [177]. Then, one needs to eliminate confusing parts [82], learn features that are robust to the specific types of noise [144] or fill in the missing parts [35].

**Evaluation.** To evaluate retrieval, we have a ground-truth such that we know which dataset shapes are related to the query [43, 70, 166]. The goal of the evaluation is two-fold: First, is the information returned from the algorithm correct? Second, does the algorithm retrieve all relevant data in the database? The measurement that focuses on the first goal of the evaluation is called *precision*, and it is the ratio of the number of relevant shapes retrieved ($TP$ so-called true positives) to the total number of shapes retrieved from the database, $prec = TP/(TP + FP)$. The denominator, the total number of shapes retrieved, is $TP$ plus any irrelevant shapes retrieved (so-called false positives $FP$). The second measurement is called *recall*, and it is the ratio of the number of relevant shapes retrieved $TP$ to the total number of relevant shapes, $rec = TP/(TP+FN)$. Thus, the denominator is the number of relevant retrieved $TP$ and relevant not-retrieved, called false negatives $FN$. Note that

precision and recall are connected,

$$TP = FN \cdot \frac{rec}{1 - rec} = FP \cdot \frac{prec}{1 - prec} \tag{2.1}$$

For example, when the algorithm is tuned to increase the precision and therefore to be more selective in accepting database shapes as TP ($FP$ decreases), this decreases the recall, as some relevant results can be missed ($FN$ increases).

To finally evaluate the retrieval, precision and recall are calculated for the **first** returned shape. Recall will be small as we now return only one shape. If the returned shape is relevant, precision is one, if not precision is zero. Then we calculate precision and recall for the first **two** shapes, then first three etc. This process creates a popular *precision-recall curve* [171, 141, 85, 21]. While the curve gives insight into the performance, one is also interested in reporting the performance as a single number. This is done by calculating the area under the precision curve (APR) [171, 141, 84, 21]. If APR is one, the algorithm works perfectly. If APR is lower, it weakens.

## 2.3 Classification

Classification (also known as class recognition or categorization) identifies a category for a new observation based on the training set where the category membership is known.

Classification algorithms are applied to a variety of problems such as recognizing objects in images [92], drug discovery [54], speech recognition [145], recognition of handwriting [36], etc. These applications motivate the research community to solve classification in an efficient way.

**Baseline.** A simple toy method to recognize the class of the unknown query 3D shape is the naive nearest-neighbor. Thus the class of the query corresponds to the class of the most similar object from the database. In this baseline method, one can simply use the result of the retrieval algorithm above.

**Challenges.** The above method does not learn anything as it performs a simple object-to-object comparison. Thus, the method expects a database that contains one example very similar to the object we want to classify. When the assumption does not hold, the algorithm does not work properly [17]. A variety of methods tackles this by searching for class specific features [181, 18], learning a relative position of the features to the object center [104], learning relations between parts [49], or by introducing hidden variables [92] that model the data.

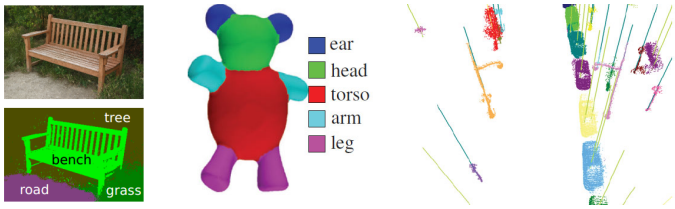



Figure 2.5: **Segmentation.** Labels of the object's elements (pixels for image, vertices/faces for 3D shapes) are estimated. From left to right: Efficient image segmentation using full-connected CRF (courtesy of [91]). Segmentation of the object parts (courtesy of [76]). Segmentation of object parts in the point cloud (courtesy of [61]).

**Evaluation.** To evaluate the classification task, we again report TP and FN as described before in §2.2. Herein, TP corresponds to correctly classified shapes, and FN to miss-classified shapes.

## 2.4 Segmentation

Segmentation estimates the class label for each basic element of the observed scene (pixel for images, voxel/vertex/face/point for 3D) as shown in fig. 2.5. It is highly related to classification as well as to the object detection [94]. While the goal of object detection is to estimate the overall location of the object, segmentation goes further to find object boundaries. It identifies the object and „segments" it out from the background.

**Baseline.** The baseline shows its link to classification as it is straightforward: Define the 3D segmentation task as classification on the level of the scene's vertices. Thus, the naive baseline learns a classifier on the set of vertices that correspond to the object against the set of vertices that correspond to the background. Then, segmentation is obtained by applying a classifier on each vertex independently [61].

**Challenges.** The above described method lacks precision for several reasons. Local parts of the objects are often similar to the background. It is important to leave the pure local point of view and incorporate dependencies between these local parts [164]. For example, should a vertex change its labeling if its neighbor has a different label? Spatial distances between vertices should not be the only consideration for measuring how vertices affect each other. Vertices

that correspond to the same object intuitively depend on each other wherever they are, even if they are very far from each other. In addition to this, detection and segmentation are connected, and they can support each other [94, 149].

**Evaluation.** The goal is to measure the portion of vertices (points) within the correctly estimated class label. Thus, similar metrics as in §2.2 are used. Precision/recall is now a portion of correctly segmented points and a portion of true labels that are recognized.

## 2.5 Datasets

Throughout the thesis, our methods and their competitors are evaluated on a variety of datasets. We now summarize these datasets.

(i) Princeton benchmark [166]. The Princeton benchmark has been one of the most popular 3D benchmarks for a long time. It consists of about 1.8K hand-made shapes (half training and half testing) and a variety of ground-truth labels at a different coarse level, such that a shape is a car as well it is a subcategory of cars, the sedan.

(ii) TOSCA [20]. This dataset consists of about 300 hand-made shapes of animals and humans in a variety of poses. We also modified this dataset to evaluate segmentation in Chapter 6.

(iii) SHREC'09. This is a dataset of 40 classes, 720 training and 20 partial query shapes from the Partial Shape Retrieval Contest [43] with complete ground-truth. The dataset presents a challenge as only a part of the object (half) is observed during test time.

(iv) KUL [85]. A simple dataset for testing purposes consisting of 94 training shapes of 8 classes and 22 query shapes.

(v) Arc3D-bikes. We took several pictures of bikes and used Arc3D software [186] to create a 3D model.

(vi) Ottawa [61]. This dataset covers about 4km$^2$ part of Ottawa city with ground-truth labels for each 3D point such as lamps, cars, poles etc. 3D data were obtained with a LIDAR scanner and the ground-truth part consists of about 7M 3D points.

(vii) Huang [70]. A dataset of at least one and half thousands shapes for each out of the three following classes: cars, planes, and chairs. The data

were crawled from the Google Warehouse [62]. Data are normalized and aligned.

# Chapter 3

# 3D SURF & Three-dimensional ISM for shape classification

Most methods for 3D shape classification focus on classifying manually generated models. However, 3D shapes obtained through acquisition techniques such as Structure-from-Motion or LIDAR scanning are noisy, contain clutter and have holes. In that case global shape features—still dominating the 3D shape class recognition literature—are less appropriate. Recently, inspired by 2D methods, researchers have started to work with local features. In line with this strand, this chapter proposes a new robust 3D shape classification method. It contains two main contributions. First, we extend a robust 2D feature descriptor, SURF, to be used in the context of 3D shapes. Second, we show how 3D shape class recognition can be improved by probabilistic Hough transform based methods. Through our experiments on shape retrieval, we show the power of the proposed 3D features. Their combination with the Hough transform yields superior results for class recognition on standard datasets. The potential for the applicability of such a method in classifying 3D obtained from Structure-from-Motion methods is promising, as we show in some initial experiments.

The chapter was published in [85] and it is organized in the following way. A 3D extension to SURF [14] serves as our local descriptor and is described in §3.2. This feature has proved quite effective in 2D and can now be viably computed even in 3D. In contrast to a dense or random coverage with spin images [74], a 3D interest point detector picks out a repeatable and salient set of interest

points. These descriptors are quantized and used in a Hough approach, like Implicit Shape Model (ISM) [104], which keeps the influence of each feature better localized than in a BoW approach as seen in §3.3. Our approach favorably compares to the state-of-the-art in 3D shape class recognition and retrieval as seen in §3.4.2 and §3.6. In this chapter, we assume all shapes are aligned (even if 3D SURF is rotation invariant), but we will introduce rotation invariant ISM in the following chapter 4.

## 3.1 Introduction and related work

A number of methods for 3D shape class recognition have been proposed already. So far, the dominant line of work has been to use global features, i.e. features that need the complete, isolated shape for their extraction. Examples are Fourier or spherical harmonics [87, 160], shape moments [160], shape histograms [129]. There are at least three potential problems with these global approaches.

(i) It is difficult to handle partial shapes. For instance, when an artifact has been damaged, even the most perfect scan will still only capture a part of what the original shape should have been.

(ii) Many capturing scenarios contain irrelevant, neighboring clutter in addition to the relevant data coming from the object. Global methods mix the two, jeopardizing class recognition. Some local, skeleton-based descriptions are also known to suffer from these problems (e.g. [105]).

(iii) Several classes contain deformable shapes, some parts of which may be more deformable than other more rigid parts.

Global methods are also less successful at handling intra-class variations while remaining sufficiently discriminative to noise, clutter, articulated deformations and inter-class variations. In many 3D application based on retrieval, classification and detection, all these four problems have to be addressed.

As work in 2D object class recognition has shown, the use of local rather than global features is advantageous. 2D class detection methods deal with occlusions and clutter quite successfully already. We therefore seek to apply these techniques in the 3D case as well. So far, relatively few 3D categorization methods based on local features, like tensors [117], heat kernel signatures [177], integral shape descriptors [58, 146], and scale dependent features [127] have been proposed.

Ovsjanikov *et al.* [131] extended the standard bag-of-visual words (BoW) approach of Sivic and Zisserman [171] by looking for the frequency of word

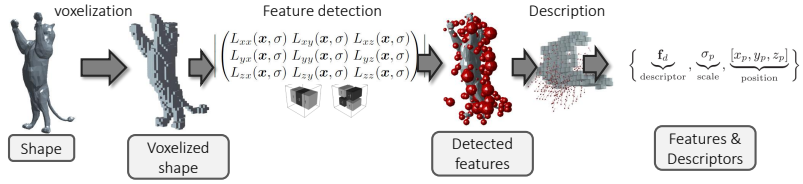



Figure 3.1: **Pipeline of the feature extraction and description**.

pairs instead of single words, called spatially-sensitive BoW. Toldo *et al.* [181] described 3D shapes by splitting them into segments, which are then described on the basis of their curvature characteristics. These descriptors are quantized into a visual vocabulary. Finally, an SVM is learnt for the actual categorization. Methods that use other information than pure shape (e.g. [61, 23]) are not considered here because we are interested in the still-common case where no other information is available.

The afore mentioned methods assume clean, pre-segmented shapes, i.e. without them being attached to a 3D 'background'. As such, these BoW approaches could suffer from the problem that the information can get buried under clutter, especially when the object of interest is small compared to this background. In 3D this difference is magnified. For instance, a statue of a person in front of a building may cover a large part of the 2D image scene, but will be tiny compared to the size of the building in 3D, where all objects appear with their actual, relative scales. In Hough transform based approaches, the process of recognition is tied up with hypothesis verification (through object localization). This means that it has higher discriminative power against clutter than BoW based approaches.

## 3.2 Shape representation as the set of 3D SURF features

For our problem of class recognition, we collected a set of shapes separated into two disjoint sets: i) training data and ii) query data. Every 3D shape is represented as a collection of vertices and triangles.

In order to describe each shape as a set of local rotation and scale-invariant interest points, we propose an extension of SURF to 3 dimensions. The method is visualized in fig. 3.1 and it has two main parts:

(i) *Feature detection* that finds local salient points (also called feature points).

(ii) *Feature description* that assigns a unique vector to each feature point such that it depends on the geometry around the feature point. Thus feature points with the similar geometry will have similar descriptors.

It is important to note, that this extension can also be seen as an adaptation of the Hessian-based spatio-temporal features by Willems *et al.* [188]. There typically a different temporal and spatial scale was applied, whereas in our case all 3 dimensions are spatial and therefore come with the same scale. On the other hand, the spatio-temporal features local frame is aligned, while we will orient the features according to the local shape. Furthermore, most of the implementation details can be reused, except the fact that the search space has now shrunk from 5 dimensions ($x$, $y$, $z$, $\sigma$ and $\sigma_{temporal}$) to 4 dimensions (avoiding the $\sigma_{temporal}$). We will now describe details of 3D SURF features.

### 3.2.1 Feature detector

The extraction of the 3D feature points, the detector, is as follows: First, we voxelize a shape in a volumetric 3D cube using the intersection of faces with the grid-bins, after which each shape is uniformly scaled to fit the cube with 40 bins along each axis. Next, we compute a saliency measure $S$ for each grid-bin $\vec{x}$ and several scales $\sigma$. As in Bay *et al.* [14], scale space is divided into octaves. Each octave is further subdivided into a constant number of scale levels. It is important that the difference between neighboring scale levels in each octave is higher than in the previous octave [14] as illustrated in fig. 3.2. Thus, the algorithm can focus on details in the finer scale levels (octave) because the neighboring scale levels will not change as much as for coarse scale levels. 3D SURF was performed over three octaves. Feature points correspond to the local extremes of the Hessian filter responses [14, 188]. We define $S$ as the absolute value of the determinant of the Hessian matrix $\mathtt{H}(\vec{x},\sigma)$ of Gaussian second-order derivatives $L(\vec{x},\sigma)$ computed by box filters (fig. 3.3),

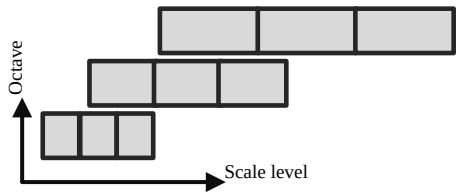



Figure 3.2: **Illustration of octaves and scale levels.** Scale space is subdivided into octaves as in [14, 188]. Each octave focuses on the different level of detail.

$$S(\vec{x},\sigma) = \left|\mathtt{H}(\vec{x},\sigma)\right| = \left| \begin{pmatrix} L_{xx}(\vec{x},\sigma) & L_{xy}(\vec{x},\sigma) & L_{xz}(\vec{x},\sigma) \\ L_{yx}(\vec{x},\sigma) & L_{yy}(\vec{x},\sigma) & L_{yz}(\vec{x},\sigma) \\ L_{zx}(\vec{x},\sigma) & L_{zy}(\vec{x},\sigma) & L_{zz}(\vec{x},\sigma) \end{pmatrix} \right|, \tag{3.1}$$

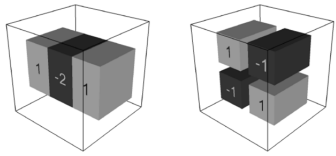



Figure 3.3: **Filters.** Two types of filters for the 3D Gaussian second ordered partial derivatives. Left figure shows filter for one direction, right figure shows filter for two directions. Courtesy of Willems *et al.* [188].

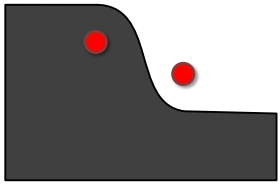


Figure 3.4: **Position of detected features.** Due to the properties of the Hessian detector, test object (in gray color) will have some features at a distance from the object (red points).

as proposed in [188]. This has as the implication that, unlike in the case of 2D SURF [14], a positive value of $S$ does not guarantee that all eigenvalues of $\mathtt{H}$ have identical signs. Consequently, not only blob-like signals are detected, but also saddle points. In addition, these detections appear in local maximum as well as minimum. Thus they can be found even at a distance from the surface of the object (fig. 3.4). Finally, unique features are extracted from the volume using non-maximal suppression [188].

### 3.2.2 Feature descriptor

In a second stage, a rotation and scale-invariant 3D SURF descriptor is computed around each interest point (fig 3.1). The process of calculating feature description is twofold: First, the orientation of the feature point is estimated and then, once the orientation is known, we use this orientation to get the local coordinate frame of the feature point. Second, the descriptor of the feature point is computed given its local coordinate frame. We now describe these two parts in detail. We uniformly sample popular Haar-wavelet responses [14, 188] in 3D space along all 3 axes within a distance $3 \times \sigma$ from each feature. These Haar-wavelet responses correspond to the approximate derivatives for each direction [14] (fig. 3.5). Next, each response is weighted with a Gaussian centered at the interest point, in order to increase robustness to small changes in position. Each weighted response is plotted in the space spanned by the 3 axes. We sum the response vectors in all possible cones with an opening angle of $\pi/3$ and define the direction of the longest resulting vector as the dominant orientation. However, instead of exhaustively testing a large set of cones uniformly sampled over a sphere, we approximate this step by putting a cone around each response. After the dominant direction has been obtained, all responses are projected

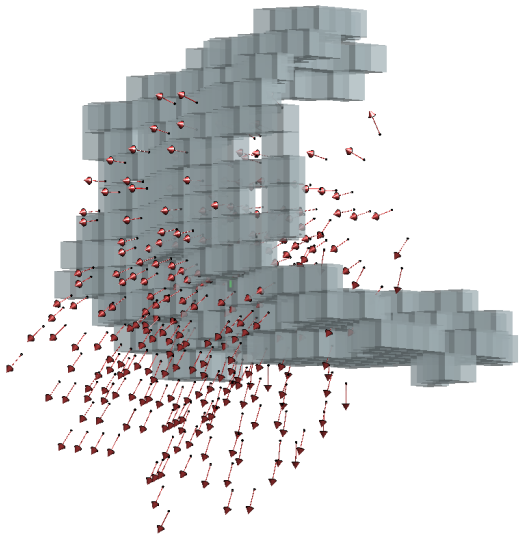


Figure 3.5: **Wavelets around the feature point.** The discretized part of a shape close to a feature point is shown as cubes. Grid-bins with calculated wavelets are visualized as red arrows. Note the relative direction of wavelet to the shape.

along this direction after which the second orientation is found using a sliding window [14]. The two obtained directions fully define the local frame. Defining a $N \times N \times N$ grid around the feature and computing the actual descriptor is implemented as a straight-forward extension of the 2D version. At each grid cell, we store a 6-dimensional description vector of Haar wavelet responses as in [188]. In the rest of the chapter, we assume $N = 3$.

For the feature $k$ of the shape $m$ we maintain a tuple of associated information which consists of feature position, feature scale and the descriptor, formally:

$$\left\{ \underset{3\times1}{\mathbf{p}_{mk}} \;,\; \sigma_{mk} \;,\; \underset{162\times1}{\mathbf{d}_{mk}} \right\}, \tag{3.2}$$

where $\mathbf{p}_{mk}$ represents the 3D position of the feature point, $\sigma_{mk}$ is the scale of the feature point and $\mathbf{d}_{mk}$ is the 162-dimensional 3D SURF descriptor vector, where $162 = 3 \times 3 \times 3 \times 6$. The descriptor is often replaced by the closest visual word, $c_{mk}$.

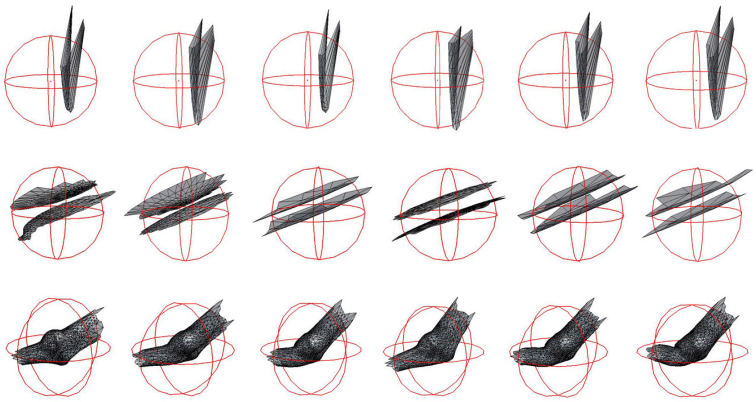


Figure 3.6: **3D visual words.** Each row shows some features that belong to the same visual word. The feature is located in the center of each sphere, while the sphere's diameter represents the feature scale. Each surface is shown normalized w.r.t. the scale of the feature.

## 3.3 Implicit Shape Model for 3D classification

In order to correctly classify query (test) shapes, we need to assemble a model of each class based on the local 3D SURF features, and define a ranking function to relate a shape to each class. Based on the information acquired during training, the quantized version of each 3D SURF descriptor (visual word) on a test query shape then casts weighted votes for the location of the shape center for a particular class, as will be seen in §3.3.2. Depending on whether the query shape's center is already known, the above information is used for classification in two ways as outlined in §3.3.3.

### 3.3.1 Visual Words Construction

Our class model combines two measurements on the local features: i) an appearance; and ii) relative position. The appearance is calculated using the BoW approach already introduced in §2.1.4 and we briefly overview it here again. In BoW, features from the training set are clustered to form visual words. Each visual word is a cluster center from the set of clusters. All features are then represented as the index to the closest visual word instead of the high dimensional descriptor vector. Features that correspond to the same visual word, should have similar descriptor, so their local neighborhood should be similar too (fig. 3.6).

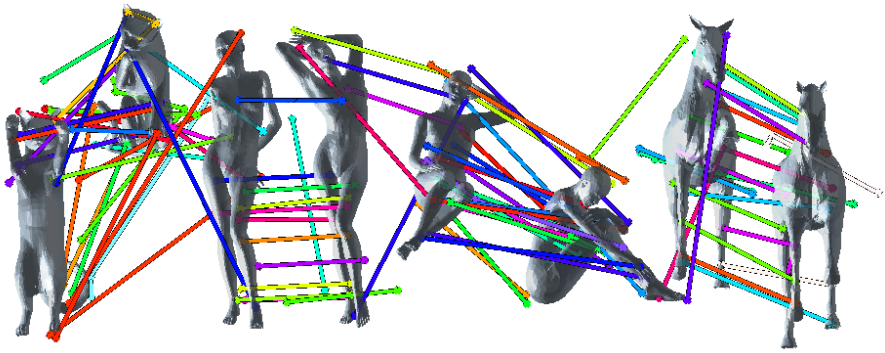


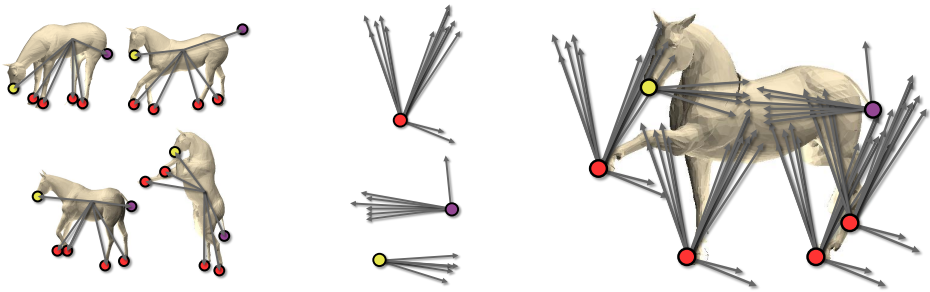
Figure 3.7: **Visual vocabulary based correspondences between 3D shapes.** Corresponding visual words are highlighted by colored lines, where each color reflects the id of the specific visual word.

Following standard practice [171, 73] in large-scale image searching, we set the number of visual words (clusters) to 10% of the total number of features in our training set. In practice, this yields a reasonable balance between mapping similar shapes to the same visual word (fig. 3.6), while ensuring that features that are assigned the same word are indeed likely to correspond (fig. 3.7).

### 3.3.2 Learning and Weighting Votes

Rather than storing a shape for each class, the ISM-based methods keep track of where a visual word $c$ would be located on a shape of class $\ell$ relative to center of the 3D shape [104, 101]. This information – the collection of visual words and offsets from shape centers – is assembled from the training set, and stored along with the visual words themselves (fig. 3.8). In the training part of the 3D ISM model, we therefore associate each visual word $c$ with a list of votes representing the relative position of $c$ to the center of class $\ell$. Each of those being generated from a feature position and defined by: i) the feature's class $\ell$; ii) its relative position to the shape center $[x^*, y^*, z^*]^\mathsf{T}$; iii) and its scale $\sigma^*$. Each word may therefore cast votes for multiple classes on many scales. Words may also cast multiple votes for the *same* class (fig. 3.9), because there may be multiple features on a shape associated with the same visual word.

In the test part, each feature casts several votes where it expects the center of the class is. Suppose now that a visual word $c$ was already observed at position $[x^*, y^*, z^*]^\mathsf{T}$ (relative to the center) during the training. Then, if a test shape contains a feature at location $[x, y, z]^\mathsf{T}$ with scale $\sigma$ that is assigned to visual



(a) Training data (b) 3D ISM model (c) Classification

Figure 3.8: **Illustration of 3D ISM model.** (a) Features are highlighted by the color points that represent different visual words. (b) For each visual word, we store its relative position to the object center. (c) In the test part, visual words cast votes where they expect the center of the object.

word $c$. That feature will cast a vote into object center location, $\boldsymbol{\lambda}$, for a shape of class $\ell$ at

$$\boldsymbol{\lambda} = \left[x - x^*(\sigma/\ \sigma^*), y - y^*(\sigma/\ \sigma^*), z - z^*(\sigma/\ \sigma^*), \sigma/\ \sigma^*\right]^{\mathsf{T}}, \qquad (3.3)$$

with relative shape size $\sigma/\ \sigma^*$. This process is illustrated in fig. 3.8(c). If the test query shape exactly matches a training shape, the votes associated with that training shape will all be cast at the test query shape's center, yielding a strong cluster of votes for the match. On the other hand, the votes associated with a training shape from a different class will get scattered around, because the spatial arrangement of features (and therefore visual words) will be different (fig. 3.9).

Note that although a single assignment of features to the closest visual word is natural, it is subject to noise when cluster centers are close together. Therefore, during the training phase, each feature activates the closest word and every other word within a distance $\tau$ in the descriptor space, as in [104, 73, 142]. Parameter $\tau$ ensures that similar visual words that are located at the same position on a shape will all vote appropriately.

An issue is that different classes may have different numbers of features, and not all features discriminate equally well between classes. We account for this by introducing a pair of factors to weight the vote cast at location $\boldsymbol{\lambda}$.

(i) A statistical weight $w_{st}$, as every vote should be invariant to the number of training samples in the class,

(ii) A learned weight $w_{lrn}$ that ensure that votes correctly votes for a class center across training shapes.

We now describe these two factors in detail.

**Statistical weight:** The statistical weight $w_{st}(\ell_i, c_j)$ takes into account: i) different number of vote casts for each class; ii) total number of votes from the particular visual word; iii) a relative number of votes per detected feature point. Formally, all the vote casts by visual word $c_j$ for class $\ell_i$ are weighted by

$$w_{st}(\ell_i, c_j) = \frac{1}{n_{vw}(\ell_i)} \cdot \frac{1}{n_{vot}(c_j)} \cdot \frac{\frac{n_{vot}(\ell_i, c_j)}{n_{ftr}(\ell_i)}}{\sum_{\ell_k \in \mathcal{L}} \frac{n_{vot}(\ell_k, c_j)}{n_{ftr}(\ell_k)}}, \tag{3.4}$$

where the different numbers $n$ are determined from the training set. For instance, $n_{vot}(c_j)$ is the total number of votes from the visual word $c_j$, $n_{vot}(\ell_i, c_j)$ is the number of votes for class $\ell_i$ from $c_j$, $n_{vw}(\ell_i)$ is the number of visual words that vote for class $\ell_i$, $n_{ftr}(\ell_i)$ is the total number of interest points in training part of the class $\ell_i$. $\mathcal{L}$ is the set of all classes. The first factor makes every class invariant to the number of visual words in its training set, while the second normalizes for the number of votes each visual word casts. The final term reflects the probability that $c_j$ votes for class $\ell_i$ as opposed to some other class.

**Learned weight:** Additionally, motivated by Maji and Malik's *et al.* [110] work, we normalize votes on the basis of how often they vote for the correct training shape centers. We define $\lambda_{ij}$ as the vote cast by a particular instance of the visual word $c_j$ on a particular *training* shape of class $\ell_i$; that is, $\lambda_{ij}$ records the distance of the particular instance of visual word $c_j$ to the center of the training shape on which it was found. We now apply this vote to *every* instance of visual word $c_j$ on *every* training shape in class $\ell_i$, and compute a Gaussian function of the distance between the center position voted for and the actual center. This scheme puts more emphasis on features with voted positions close to that actual center.

For every vote $\lambda_{ij}$ of shape class $\ell_i$ and casted from visual word $c_j$, our goal is to obtain one value summarizing the statistics of distances to shape centers:

$$w_{lrn}(\lambda_{ij}) = \text{median}\left(\left\{ e^{-\frac{d_a(\lambda_{ij})^2}{\sigma^2}} \;\middle|\; a \in A \right\}\right), \tag{3.5}$$

where $A$ is the set of all features associated with word $c_j$ on a shape of class $\ell_i$ and $d_a(\lambda_{ij})$ is the Euclidean distance as just defined. We use a standard

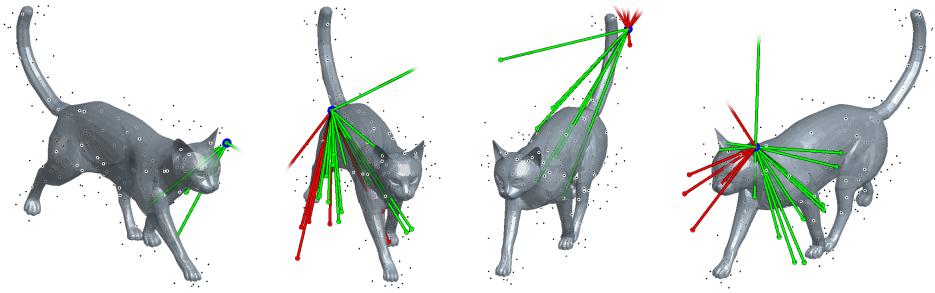


Figure 3.9: **Votes cast from four features on a cat shape instance.** All detected features are visualized as small black dots. and votes are shown as lines starting from the feature (marked blue). The votes from a toy ISM model were learned from six shapes of the cat-class (visualized as green lines) and six shapes of the flamingo-class (red lines). Note six votes cast from the cat's tail in a direction to the center of the object. Votes cast from less discriminative patches, i.e. back of the cat, sometime confuse. Thus some wrong vote casts occur. Overall, votes cast for the cat-class will obviously win against the flamingo-class.

deviation of $\sigma$ taken as 10% of the shape size, which defines the accepted amount of noise.

The final weight of the vote $\lambda_{ij}$ is the product of $w_{st}$ and $w_{lrn}$ as the correct object centers should have both weighting factors high,

$$w(\lambda_{ij}) = w_{st}(c_j, \ell_i) \cdot w_{lrn}(\lambda_{ij}). \tag{3.6}$$

### 3.3.3 Determining a Query Shape's Class

The class recognition decision for a given 3D query shape is determined by the set of 5D votes (shape center, size of the shape and class), weighted by the function $w(\lambda_{ij})$. However, we need a mechanism to cluster votes cast at nearby but distinct locations. Depending on the type of query shape, we use one of two approaches:

**Cube Searching (CS):** In the spirit of Leibe *et al.* [104], we discretize the 4D class specific search space into bins; each vote contributes to all bins based on its Gaussian-weighted distance. The selected class and shape center is given by the highest score. The space with the center having the most votes specifies the class. The principal advantage of this approach is that it does not require

a clean query shape — noisy or partial query input is handled by explicitly searching for the optimal shape center as well as the class.

**Distance to Shape Center (DC):** Unlike image queries, where the shape's center within the image is usually unknown, it is quite easy to heuristically estimate the centroid of a clean 3D shape as the gravity point of the set of faces and vertices, and use this as the shape center. Doing so can simplify class recognition and improve its robustness by reducing the search to the best class given this center. We do this by weighting each vote by a Gaussian of its distance to the query shape's center. This heuristic would work on such complete 3D shapes, which is a popular task in 3D literature [181, 131]. Obviously, the assumption that real object center coincides with the shape center is not always valid and we cannot use it for partial shapes or for the recognition of 3D scenes (with additional clutter or noise).

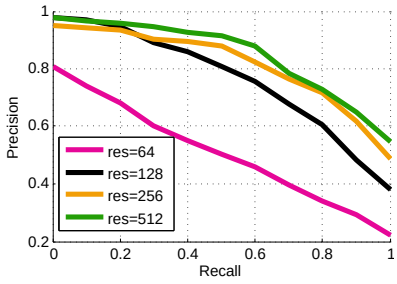
## 3.4 Evaluation of 3D SURF features

Prior to the classification experiments, 3D SURF features are evaluated. We show two experiments. First, we investigate the performance of 3D SURF features with respect to its parameters and several kinds of noise that can occur on shapes in §3.4.1. Second, in §3.4.2, we show the 3D SURF's performance against the state-of-the-art detectors/descriptors on shape classification or retrieval.

### 3.4.1 Sensitivity to parameters

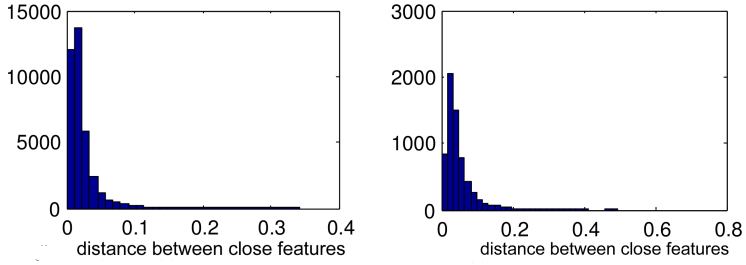
We evaluated the performance of 3D SURF with respect to the resolution of the discretization cube and threshold to detect features.

For the resolution of the discretization cube, we run shape retrieval using 3D SURF on 64, 128, 256, and 512 dimensional grids with parameters are fixed. We measured two characteristics: i) the number of detected features and ii) precision-recall curves for the retrieval task on the KUL dataset. Fig. 3.10 shows the results. As expected, a finer discretization produces more features and the performance improves. Note that SURF using a $128^3$ dimensional grid produces 17% features compared to the those of size $256^3$ and only 4% compared to the $512^3$, while the results are not significantly worse. In the rest of the experiments, we used 3D SURF with the $256^3$ dimensional grid.




| *resolution* | # of features |
|---|---|
| 64 | 7.2 |
| 128 | 60.5 |
| 256 | 353.0 |
| 512 | 1440.4 |

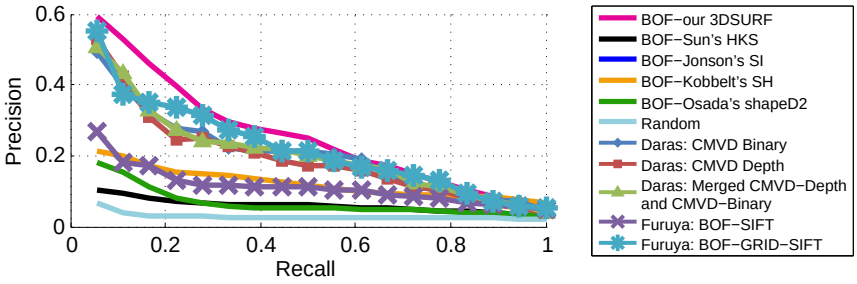
Figure 3.10: **SURF performance with respect to the resolution of the discretization cube.** Retrieval performance was computed as the average PR on the KUL dataset. Increasing the resolution of the discretized cube produces better performance due to more details on the voxelized shape. While the number of features still increase dramatically with the finer resolution, the performance improvement starts to be insignificant when the discrectization starts to be fine (256 vs. 512).



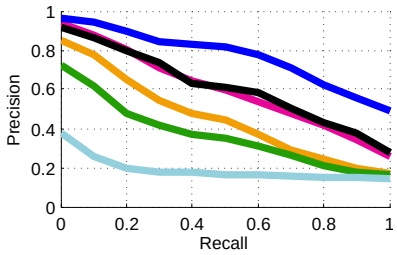

(a) (b)

Figure 3.11: **Position of detected points w.r.t. discretization.** The figure shows two histograms of distances between SURF features detected at one resolution to the closest feature points at another resolution. Shapes were normalized to have unit size. (a) Compares 128 and 256 dimensions and (b) 256 and 512 dimensions.

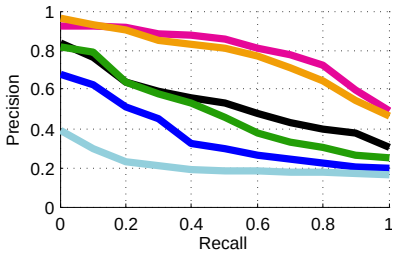
In fig. 3.11, we plot the difference between positions of the detected feature points w.r.t. the size of the discretization cube. As a significant majority of points is placed around the zero distance, it demonstrates that the SURF detector emphasizes features at the same locations when the dimension of the discretization grid varies.




(a) SHREC'09 dataset




(b) Tosca, small resolution shapes




(c) KUL dataset

Figure 3.12: **Comparison of different detectors/descriptors using the Video Google [171] retrieval approach.** The performance is measured as Precision-Recall curve. (a) The SHREC'09 Partial Shape Retrieval Contest [43] provided results which were compared with our 3D SURF approach and others. (b,c) Note that the performance highly depends on the characteristics of the shapes as the results very much depend on the dataset.

### 3.4.2 Performance of 3D SURF vs. state-of-the-art detectors/descriptors

We have presented a novel method for local features detection and description for 3D shapes and we have shown its sensitivity to certain parameters. Now, we compare the performance of our approach to that of the state of the art descriptors discussed in §2.1.4.

**3D SURF in shape retrieval.** As the task here is shape retrieval (as opposed to our classification based method in §3.3), we use 3D SURF features in the large-scale image retrieval approach of Sivic and Zisserman [171] based on BoW. First, the 3D SURF features of all shapes were quantized using the same visual vocabulary as in §3.3.1. Second, we compute the BoW vectors as histograms of occurrences of visual words on the shape. Third, using the

| | | *KUL* | | | *Tosca, small res.* | | | *SHREC'09* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| descriptor | type | #TP | #FP | perfor. | #TP | #FP | perfor. | #TP | #FP | perfor. | mean perfor. |
| Gehler *et al.* [57] | combina. | 18 | 4 | 81.8% | 17 | 2 | **89.5**% | 13 | 7 | 65.0% | **78.8%** |
| 3D SURF | local | 20 | 2 | **90.9**% | 12 | 7 | 63.7% | 12 | 8 | 60.0% | **71.2%** |
| 2.5D SURF | global | 18 | 4 | 81.8% | 8 | 11 | 42.1% | 15 | 5 | **75.0**% | **66.3%** |
| SH [87] | global | 20 | 2 | **90.9**% | 13 | 6 | 68.4% | 1 | 19 | 5.0% | **54.3%** |
| HKS [177] | local | 11 | 11 | 50.0% | 16 | 3 | **84.2**% | 2 | 18 | 10.0% | **48.1%** |
| SI [74] | local | 13 | 9 | 59.1% | 15 | 4 | **78.9**% | - | - | -% | –% |
| HoughOct [2] | global | 16 | 6 | 72.7% | 12 | 7 | 63.7% | 0 | 20 | 0.0% | **45.5%** |
| shapeD2 [2] | global | 11 | 11 | 50.0% | 7 | 12 | 36.8% | 0 | 20 | 0.0% | **12.4%** |

Table 3.1: **Results for different shape descriptors using the classification method of Toldo *et al.* [181].** On the *KUL* dataset, we found that 3D SURF and SH outperform all other descriptors. 2.5D SURF and HoughOct perform well also. While HKS performs badly on KUL, it is the winner with a 84% performance on the Tosca dataset (representing clean shapes in different poses). Spin Images perform well in this case, but their computation time (on average >15min for one shape) is the cause for presenting no results on the challenging SHREC'09 dataset. For SHREC'09, only 3D SURF and 2.5D SURF recognize the majority of the test shapes. Other methods fail for this experiment.

BoW, every shape model is represented as the normalized tf-idf vector [157] preferring the discriminative visual words. Finally, similarity between two 3D shapes is estimated as the similarity between their tf-idf vectors. We now show performance of such BoW using 3D SURF on three datasets.

Fig. 3.12(a) presents our results together with results from the SHREC'09 Partial shape retrieval contest [43]. Note that 3D SURF features outperform SIFT descriptors calculated on rendered range images, in similar BoW frameworks.

Fig. 3.12(b,c) shows the retrieval performance on two additional datasets. As the main result, we observed high sensitivity of all descriptors to the dataset type, i.e. SI [74] outperforms all methods in the Tosca dataset, while it gives the worst results on the KUL dataset. We skip SI's evaluation on the SHREC'09 dataset as the implementation we had did not calculated SI descriptors successfully on this set of shapes.

We also observed (on shapes from 1.2K-65K faces and 0.6-33K vertices) that our method is faster than other local descriptors. On average, 3D SURF takes 20.66s, HKS [177] 111.42s and SI [74] more than 15mins. The experiment was performed on 4xQuad Core AMD Opteron, 1.25Ghz/core.

**3D SURF in shape classification** We also run the proposed 3D SURF detector and descriptor against its state-of-the-art competitors on the task of shape classification. As the classification algorithm we used Toldo *et al.* [181] in the following way that it also allows to compare local descriptors against global. For

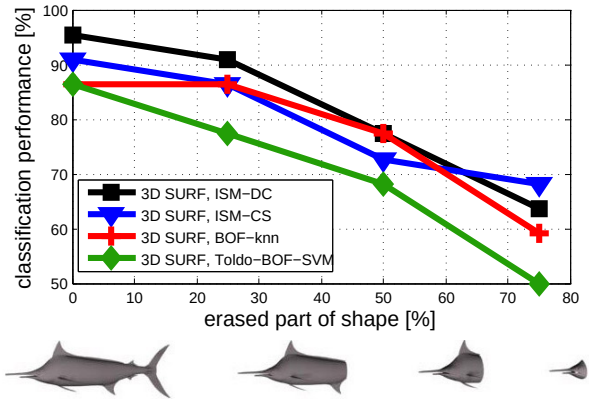



Figure 3.13: **Sensitivity of 3D classification to missing data.** The classification performance is plotted as the shape is increasingly cropped. See the fish example on the bottom row. We found that our approach outperforms knn as well as Toldo's [181] SVM method.

local descriptors, each shape is again represented as BoW vector of normalized visual words occurrences. For global descriptors, no BoW is needed because the shape is represented as a single global descriptor. On training data we learn a multi-SVM like Toldo *et al.* [181] and we measured the classification performance on the test data.

Results are shown in tab. 3.1. Again, we see that the dataset type affects results significantly. For example, while 2.5D SURF performs the best on SHREC'09, it suffers on the nice clean shapes in Tosca. HSK and SI are winners on this Tosca dataset, while they perform badly on other datasets. This is not surprising as HKS and SI are sensitive to the surface geometry, this is an advantage when the shape is clean, but it introduces errors in the presence of a noise. We also included the method of Gehler *et al.* [57] that combines **all** descriptors, which improves results. Overall, 3D SURF obtained best mean performance out of all the compared descriptors.

## 3.5 3D SURF and ISM for 3D classification

Here we apply two versions of our method (ISM, §3.3) for shape classification that is described in §3.3.3. These methods are Cube-searching (ISM-CS) and Distance to the shape center (ISM-DC). These versions of our ISM model are compared against the following:

(i) **BoW-knn:** Encouraged by the good results of the 3D shape retrieval algorithm in §3.4.2, we use the retrieval as one competitor. The test query shape is assigned to the most commonly occurring class among the best $k$-retrieved training shapes in a nearest-neighbor classification approach. The parameter $k$ was learn to optimize the classification performance of the training shapes. The shapes are represented by normalized tf-idf vectors and $L_1$ metric is used to measure their similarity. Tf-idf [171] re-weights every visual word in the BoW vector based on how often it was seen in the training data. Visual words that appear at every class are less discriminative than visual words that were seen just for one particular class.

(ii) **Toldo-BoW-SVM:** This is our implementation of Toldo *et al.* [181], where BoW vectors are computed on the training data. Then, the multi-class SVM classifier of [28] is trained on the BoW vectors to predict the class label of the test query shapes. The kernel function is defined in terms of histogram intersection as in [181].

In this chapter, 3D assume all shapes aligned (even if 3D SURF is rotation invariant). Shapes in KUL and Princeton dataset are not aligned, thus further incorporation of the rotation invariance (chapter 4) improves the results. However 3D SURF and 3D ISM are also scale invariant, datasets do not include shapes in different scales.

First, we investigate the sensitivity of these classification methods with respect to occlusions. Fig. 3.13 shows the performance of the methods in the presence of occlusion on the KUL dataset. As expected, ISM-DC gives the best results for complete models as it expects centeroid in shape's center. While ISM-DC is outperformed for partial shapes as the real center is not coinciding with the shape center. Thus, the performance of ISM-CS outperforms all methods when more than 50% of the test query shape is missing.

Table 3.2 summarizes all results on standard datasets of 3D shapes. Here, we measured the performance of the classification methods on several datasets. Our approach using the Hough voting gave the best average performance (last column in Table 3.2). The Princeton dataset is the most challenging and although all methods gave similar results, we outperform the others. This dataset has very high variation among its 3D models e.g. the animal class contains widely varying models of 'ant' and 'fish'. For an SVM to learn a good classifier, we need a good non-linear kernel which has learnt such differences well. In such cases, non-parametric nearest-neighbor classifiers have a natural advantage.

The SHREC'09 dataset, previously used for the retrieval of partial queries, is now used for classification. ISM does not perform well as this method needs a


Figure 3.14: Samples of query shapes from the state-of-the-art datasets.

| | *Princeton* | | | *Tosca+Sumner* | | | *SHREC'09* | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| method | # TP | # FP | perfor. | # TP | # FP | perfor. | # TP | # FP | perfor. | avg. perf. |
| ISM | **529** | **378** | **58.3%** | **56** | **1** | **98%** | 8 | 14 | 40% | **65.4%** |
| BoW-knn | 491 | 416 | 54.1% | **56** | **1** | **98%** | 7 | 13 | 35% | **62.4%** |
| BoW-SVM | 472 | 435 | 52.0% | 41 | 16 | 72% | **12** | **8** | **60%** | **61.3%** |

Table 3.2: **Results of classification.** Proposed approach beats k-nn and SVM in most cases.

relatively large number of training examples [104, 101] which is not satisfied in this case.

We conclude that our ISM based method beats k-nn and SVM in most cases and gave the best average performance.

### 3.5.1 3D shape classification of reconstructed real life scenes

As a final note, it is interesting to investigate the relative roles 2D and 3D object class detection could play in real-life. We carry out a small experiment to see whether 3D detection would really offer an added value.

Given many images taken in uncontrolled conditions around a real object, state-of-the-art methods such as the Arc3D web-service [186] can be used to extract a dense 3D model from the captured images. Such object models exhibit varying amounts of noise, holes and clutter from the surroundings, as can be seen from the examples (fig. 3.15). For each class in fig. 3.15 we reuse the 3D ISM models trained on datasets of the SHREC'09 (for bike and plant classes), Tosca+Sumner (for woman) and KUL (for cube and people). We also used 2D Felzenszwalb detectors [49] trained on data from the PASCAL'08 datasets for bikes, potted plants, and pedestrians. As shown in the fig. 3.15, a small test was run, where 3D reconstructions were produced from images for an instance of each of the 6 objects. In each of these cases, the classification using 3D ISM was successful, while SVM based method of Toldo *et al.* [181] failed in all cases. As to the 2D detectors, the bike was found in 12 out of the 15 images, the

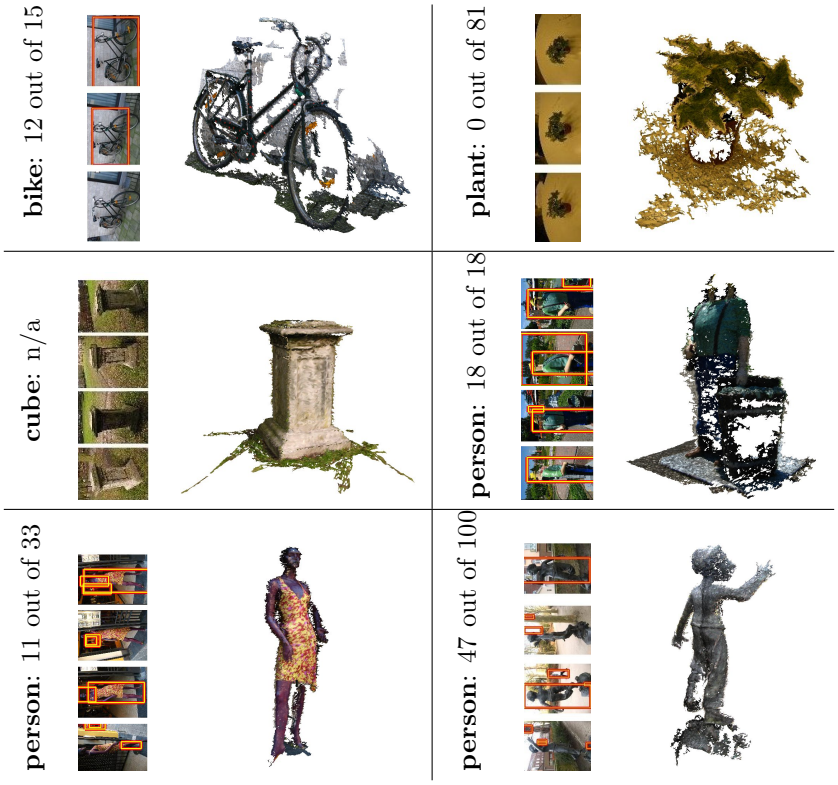



Figure 3.15: **3D class recognition from the set of images.** For each sample: correctly recognized class using 3D ISM, the number of correctly recognized objects in images using the method of Felzenszwalb *et al.* [49] (the best for PASCAL'08), samples of detection results are highlighted by squares, and the reconstructed shape by Arc3D [186].

potted plant in none of the 81 images, and the person in 47 out of the hundred. This would indicate that given a video images input, a single 3D detection into the images could be more effective than 2D defections in separate images. But issues concerning 2D vs. 3D detection need to be explored further.

## 3.6 Conclusion

In this chapter, we introduced 3D SURF features in combination with the probabilistic Hough voting framework for the purpose of 3D shape class

recognition. This work reaffirms the local feature direction taken by recent research in 2D class detection, but thereby deviates rather strongly from traditional 3D approaches, which are often based on global features. Only recently some first investigations into local features combined with BoW classification were made.

We have demonstrated through experiments the power of proposed 3D SURF features (§3.2), and then the combined power of the features and their spatial configuration (§3.5). This approach outperforms existing methods and both aspects seem to play a role in that.

So far, we have not mentioned rotation problems in the ISM framework that occur in 3D. These issues are going to be discussed in the following chapter.

# Chapter 4

# Rotation invariant ISM

The extension of ISM [104] into the third dimension (Chapter 3) introduced several challenges such as higher dimensional space, different features, a different normalization etc. While these differences were already discussed in the previous chapter, in contrast to 2D, the object is rarely observed in a canonical frame of reference with respect to its orientation (or scale). For example, pedestrians are often standing or sitting, a car is usually on the road. Then the orientation of the object is fixed to the coordinates of the image and 2D methods often assume this fixed orientation of objects [104, 101, 142]. This is not the case in 3D [20] and one has to take into account that objects are not aligned. Thus, we now present an extension of our Hough-voting based method for object classification using local 3D features, that are orientation invariant.

The chapter was published in [86] and it is organized in the following way. We discuss related work in §4.1. Then, we quickly give an overview of the relevant parts of the Hough-based approach in §4.2. We show how orientation invariance can be incorporated in §4.3. Finally, the proposed methods are evaluated and compared in §4.4.

## 4.1 Introduction & Related Work

We focus on the problem of unknown orientation and, to a lesser extent, scale. In contrast to much of the 2D object class detection work, which is viewpoint specific, an important advantage of having 3D information ought to be that the class detection is less viewpoint dependent. This requirement is strengthened

further by the fact that the orientation under which 3D samples are presented to the recognition system does not necessarily correspond to any viewpoint under which these samples were originally captured (as often 3D scans are integrated first, for instance). Thus, even if features are invariant under rotation and scale, it is equally important that all further steps along the class recognition pipeline are invariant as well. Again, in contrast to many earlier papers in 3D shape retrieval, we will not assume that the models are pre-segmented, complete and noiseless [74, 87, 58, 177].

An obvious way of realizing invariance of the entire pipeline is to use global invariant features, in which case the remaining part is sheer classification. Starting from invariant, local features, another obvious way of keeping that invariance further on during detection is by following a bag-of-features (BoF) approach. Nice examples are the work of Toldo *et al.* [181] or Shape Google [131]. In the latter case, co-occurrences of feature pairs are counted, leading to an improved performance and a weak form of information about feature configurations. The danger of spurious detection can be much larger than in 2D however, because there is no natural limitation to the size of a 3D model as there is on an image. The amount of background clutter in 3D can be substantially larger than that in an image. One may want to search for objects in large 3D city models for instance. Johnson and Hebert [74] bring in stronger information about geometric feature configurations as they estimate transformations between query and model shapes through a RANSAC-like approach. Funkhouser and Shilane [52] follow a similar pattern of forming groups of consistent correspondences, but increase the efficiency of this grouping. A caveat is that strict configuration testing works for specific shape matching, but not for class detection, due to their inherent variability.

Woodford and colleagues [189] proposed a recognition scheme of CAD models based on the ideas proposed in §3.3. They also searched for the point of maximum density in the vote space. This parallel work investigated a solution to the orientation problem. Instead of estimating the orientation of local patches and casting votes according to this local information, Woodford *et al.* [189] expanded the dimensionality of the vote space by three additional dimensions that correspond to the orientation of the object.

## 4.2 Hough transform based classification

As described in the previous chapter, there are three key parts in the Hough Transform classification process à la ISM [104]. First, a class model is learnt from a set of training shapes. In the second stage, the learnt class models are used to generate hypotheses of probable class instances for a query sample. In the final stage, this hypothesis space is searched for the strongest hypothesis.

A shape is described by a set of characteristic, locally extracted features. We use 3D SURF features. Several approaches [104, 171, 131, 181], including ours, then proceed by clustering all feature descriptors, where each cluster center represents a visual word. The visual words together with the information about their contributing features represent the model for a class.

When looking for a certain class of the query shape, the model for this class is used. Using the visual word and that model a list of relative positions are obtained, suggesting where the object center may be. If the features behind the visual word in the model were observed at relative positions $[x^*, y^*, z^*]^\top$ (with respect to the object center) and the current feature is found at $[x, y, z]^\top$, then for each model feature the vote cast for the object center is generated at position,

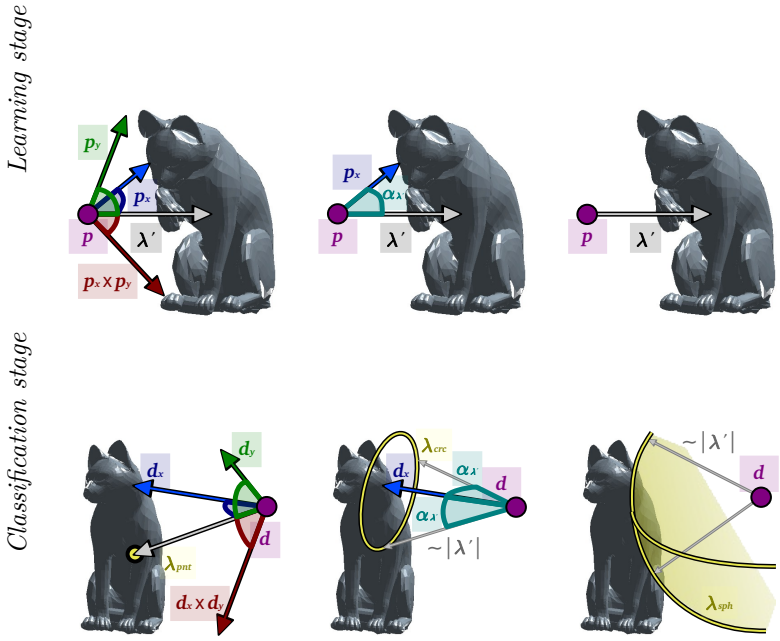
$$\boldsymbol{\lambda} = [x - x^*(\sigma/\ \sigma^*), y - y^*(\sigma/\ \sigma^*), z - z^*(\sigma/\ \sigma^*)]^\top, \tag{4.1}$$

for the specified class, and for the relative scale $\sigma/\ \sigma^*$ with $\sigma$ the scale of the shape feature observed in the training data and $\sigma^*$ the scale of the current feature. Thus, voting takes place in 4D spaces, one space for every class.

In order to achieve good results, earlier works [104, 101, 110] pointed out that it is important to normalize the votes, i.e. to assign variable weights to them. We have proposed [85] to use a two-stage normalization of a statistical weight $w_{st}$ and a learnt weight $w_{lrn}$ as described previously in §3.3.2. Votes are accumulated in a discretized, class-specific 4D space. The most likely class and its localization and scale must be obtained by finding the maximum across all classes.

## 4.3 Rotation invariant voting

The aforementioned voting scheme (presented in §4.2) will be referred to as *Rotation variant* voting as it uses fixed-directional voting (as described in (§4.1). Obviously, this is very sensitive to object orientation and the robustness of feature orientation detection. We now investigate how variations in sample and feature orientation can be handled through various voting schemes.




(a) *Point voting* (b) *Circle voting* (c) *Sphere voting*

Figure 4.1: **Illustration of rotation invariant approaches for Hough Transform based classification:** Assume, the query feature $d$ and the training feature $p$ are assigned to the same visual word. Vote $\boldsymbol{\lambda}'$ from $p$ to the corresponding training shape center can be used at query time in the following schemes: (a) **Point voting:** The query vote $\boldsymbol{\lambda}_{pnt}$ can be generated by rotating $\boldsymbol{\lambda}'$ according to the orientation of a feature-centric co-ordinate system $(\mathbf{d}_x, \mathbf{d}_y)$, corresponding versions of which are detected at training and query time. (b) **Circle voting:** When only one dominant direction $\mathbf{d}_x$ is reliably detected for feature $\mathbf{d}$, the locus of votes is a circle defined by relative angle $\alpha_{\boldsymbol{\lambda}'} = angle(\boldsymbol{\lambda}', \mathbf{p}_x)$. (c) **Sphere voting:** Votes can be cast uniformly on a sphere of radius $|\boldsymbol{\lambda}'|$ in the absence of any orientation information.

**Point voting.** Within a class-specific voting space, every feature generates directed votes for the hypothetical shape centers and shape sizes (eq. 4.1), i.e. the votes go to a discrete set of points. What has been discarded so far is the orientation of the shape. The vectors connecting features and centers were assumed to have fixed components. Yet, if the shape would rotate, these vectors should actually rotate with it.

One obvious way of achieving this is to vote relative to a co-ordinate frame attached to a feature, if this feature (and its coordinate system) rotate with the shape. Some feature detectors (e.g. [117, 188]) can achieve this by computing

local reference directions 3.2. The same Hough-voting approach can then be used, as long as voting happens in relative instead of absolute directions.

We will now describe this type of voting formally. Consider a training feature $p$ whose position to its object center is given by $\boldsymbol{\lambda}'$. Let $d$ be a feature on the query shape, assigned to the same visual word as $p$. We want to find the vote $\boldsymbol{\lambda}$ generated by the query feature. Given two reference directions $\{\mathbf{d}_x, \mathbf{d}_y\}$, the rotation invariant co-ordinate system of the point $d$ is defined by,

$$\mathsf{R}_d = \begin{bmatrix} \mathbf{d}_x & \mathbf{d}_y & \mathbf{d}_x \times \mathbf{d}_y \end{bmatrix}. \tag{4.2}$$

The relative vote $\boldsymbol{\lambda}_{pnt}$ generated by query feature $d$ (using training feature $p$) is then rotated according to its co-ordinate frame:

$$\boldsymbol{\lambda}_{pnt} = \mathsf{R}_d \mathsf{R}_p^{-1} \boldsymbol{\lambda}' \sigma_d / \sigma_p. \tag{4.3}$$

Fig. 4.1(a) illustrates the rotation of this vote. Local surface symmetry and noise on the shape may render the determination of a unique co-ordinate frame fragile. The alternative schemes that follow can mitigate this problem.

**Circle voting.** Often, despite difficulties in extracting an entire feature co-ordinate system, one dominant direction can still be reliably determined (e.g. when the local surface normal is well defined but the surface is planar or spherical). The reliable feature direction restricts the location of the object centre to a specific circle, as now derived.

Let us denote the reliable direction as $\mathbf{p}_x$ (starting from feature $p$). During training, the angle $\alpha_{\boldsymbol{\lambda}'}$ subtended by the vector to the object center $\boldsymbol{\lambda}'$ and the feature's dominant direction $\mathbf{p}_x$ at the feature position is stored in addition to existing information. For query feature $d$, the same angle must be subtended between the query feature's dominant direction $\mathbf{d}_x$ and the vote vector for the query object's center $\boldsymbol{\lambda}$. The locus of the query object's center is thus constrained to a circle defined by:

$$\boldsymbol{\lambda}_{crc} = C + r\cos(t)\mathbf{u} + r\sin(t) \cdot \mathbf{n} \times \mathbf{u}, \tag{4.4}$$

where $\mathbf{n} = \mathbf{d}_x / \|\mathbf{d}_x\|$, $\mathbf{u}$ is any unit vector perpendicular to $\mathbf{n}$, circle center $C = d + \mathbf{n}\|\boldsymbol{\lambda}'\|\sigma_d/\sigma_p \cos(\alpha_{\boldsymbol{\lambda}'})$, and radius $r = \|\boldsymbol{\lambda}'\|\sigma_d/\sigma_p \sin(\alpha_{\boldsymbol{\lambda}'})$. The above can be visualized as in fig. 4.1(b) and fig. 4.2 shows examples of circle votes.

**Sphere voting.** When it is not possible to reliably estimate even a single direction of the feature's co-ordinate frame, then the alternative is to vote for all possible centers at the prescribed distance from the feature, i.e. on a complete sphere (fig. 4.1(c)), instead of restricting voting to a point or a circle. Examples of sphere votes from some features are presented in fig. 4.2.


Figure 4.2: **Votes from selected features.** Votes for the centre of the face class are visualized in green, for the dog class in red, and for the woman class in blue. The figure shows example for relatively small numbers of votes as the ISM model was trained from $\sim 5$ shapes for each class in this experiment.

## 4.4 Evaluation

We now evaluate the presented methods for rotation invariant classification. We show the Hough Transform classification results, for which the parameter settings are described in §4.4.1. We then show and discuss the experimental comparison of the rotation invariant voting approaches in §4.4.2. Finally, we compare the effectiveness of different descriptors in combination with the classification method of Toldo *et al.* [181]. This also allows for a comparison with the proposed ISM pipeline.

### 4.4.1 Setup

For our method we use 3D SURF features (§ 3.2). To make them rotation invariant, we look for uniquely identifiable directions around the feature point. The first direction, $\mathbf{p}_x$ as defined in §4.3, is extracted in the same manner as 3D SURF features [188] get their orientations: Around the feature point, wavelets are calculated along three orthogonal, fixed orientations. The contributions for the three directions are summed independently, within conical wedges with apex at the feature point. These sums can be considered as the 3 components of a vector. The longest vector of those vectors yields the first, 3D orientation vector. While its direction is very close to the normal [188], it is not the exact normal (fig. 3.5). The cube in which the 3D SURF descriptor is extracted has

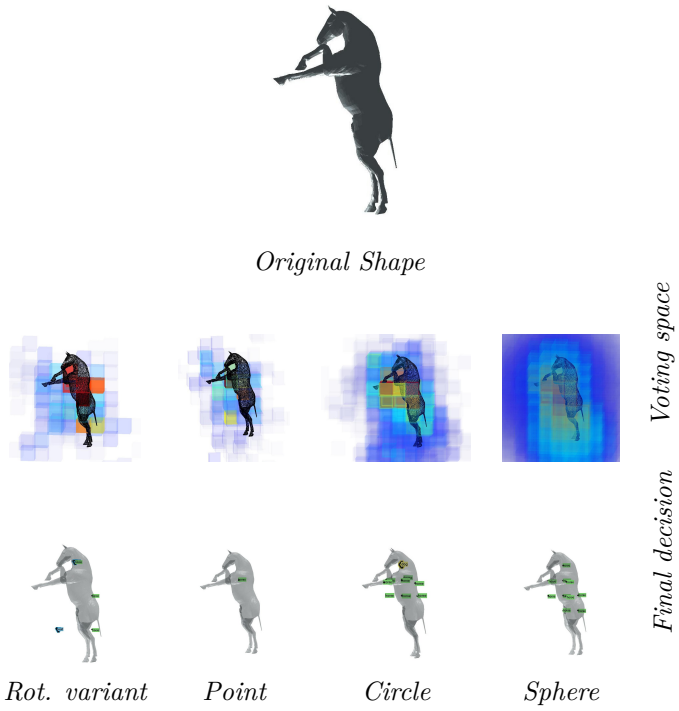



Figure 4.3: **Recognized class and location for different voting methods, for the horse shape.** Midle row (voting space): We show the density of votes. Blue-transparent corresponds to low density while red and opaqued corresponds to the high density. Bottom row(detection result): We visualize all extremes higher than 75% of the global maximum. The global maximum is found at the correct location in all cases, except for the rotation variant one, as one can visually check in the voting space. At least in this case, also the rotation variant scheme still got the class right.

one direction aligned with this vector. The cube can then still rotate about this first vector. For the extraction of the second dominant orientation, we construct a plane orthogonal to the first vector. The contrast vectors around the feature point are all projected onto this plane. Again, the vectors are summed, but now in planar wedges lying in the selected plane. The longest summed vector yields the second direction.

The resulting features are orientation and scale invariant, and exhibit robustness to noise. The result of this procedure is a set of descriptors, each associated with a 3D position and scale. SURF feature detection and description on $256^3$ dimensional grid took 20.66s on average, which is still significantly faster than

| | *Basic dataset* | | | *KUL* | | | *Tosca small. res.* | | | *SHREC'09* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| method | #TP | #FP | perfor. | #TP | #FP | perfor. | #TP | #FP | perfor. | #TP | #FP | perfor. | mean perfor. |
| Circle voting | 501 | 5 | 99.0% | 21 | 1 | 95.5% | 14 | 5 | 73.7% | 8 | 12 | 40.0% | **77.1%** |
| Crc+Sph+Pts | 718 | 2 | 99.7% | 20 | 2 | 90.9% | 13 | 6 | 68.4% | 8 | 12 | 40.0% | **74.8%** |
| Crc+Sph | 718 | 2 | 99.7% | 20 | 2 | 90.9% | 13 | 6 | 68.4% | 8 | 12 | 40.0% | **74.7%** |
| Sphere voting | 505 | 1 | 99.8% | 20 | 2 | 90.9% | 12 | 7 | 63.2% | 8 | 12 | 40.0% | **73.5%** |
| Point voting | 483 | 23 | 95.5% | 17 | 5 | 77.3% | 10 | 9 | 52.6% | 6 | 14 | 30.0% | **63.8%** |
| Rotation variant | 469 | 37 | 92.7% | 14 | 8 | 63.6% | 12 | 7 | 63.2% | 2 | 18 | 10.0% | **57.4%** |

Table 4.1: **Results of different orientation invariant approaches for the task of class recognition.** We report the number of true positives (TP), the number of false positives (FP) and the performance. We have observed that circle voting outperforms other proposals as it gives the best balance between: 1) beeing too dependent on the imprecise estimate of the local orientation (point voting) and 2) avoiding orientation by focusing on the pure distance to the object center (sphere voting).

the time required for most alternative such feature types. The calculations were performed on shapes of 1.2K-65K faces and on a 4xQuad Core AMD Opteron, 1.25Ghz/core. We detected 354 features on average for a single shape.

During the clustering part, we perform the approximate K-means algorithm of Muja *et al.* [121] to build a vocabulary with 10% of the total number of features as the number of clusters, following standard practice in retrieval/classification [171, 104].

For the Hough type voting, the voting space has been discretized into 15 bins for the 3 spatial as well as the one scale dimension. The resolution is observed to be an important parameter that influences not only the computational time, but also the performance. Choosing an adequate resolution for the discretization of the voting space is important in order to keep accuracy intact.

### 4.4.2 Experimental Results

The classification results presented in this section are summarized in tab. 4.1.

First, we perform classification on the controlled *Basic* dataset. The result is shown in the first column in tab. 4.1. *Circle* and *Sphere* voting perform well, while even the *Rotation variant* method only misclassified 7.3% of test shapes. As the dataset is designed to only test orientation robustness and not adaptability to deformation, this is an important test.

The same experiment on the *KUL* dataset, second column, shows significant improvements using rotation invariant approaches.

The *SHREC'09* dataset, which is the most challenging, was evaluated in the fourth column. It shows a significant improvement with rotation invariant methods, due to the widely varying orientations and the noise and incompleteness of the shapes. *Rot. variant* correctly classifies only 10% of the shapes, while the best rotation invariant result boosts the performance by up to 40%.

On the flipside, the *Tosca* dataset results fail to improve with the use of rotation invariance. Only circle voting brought some improvement.

## 4.5 Conclusion

In this chapter, we have presented an automatic rotation invariant method for 3D shape location and classification. Our contribution has been to integrate orientation handling into the Hough Transform based voting approach. We have investigated several possible methods to achieve this and we have compared their results on popular 3D shape datasets. We have observed that simple vector-based point voting performs badly in cases of symmetry or rotation. This behavior is caused due to these two reasons: First, symmetry of the shape yields ambiguity of where to cast votes. Second, it is impossible to estimate the rotation invariant local frame for spike like local patches (i.e. a tail-like geometry), therefore the process of casting votes cannot be robust. Thus some level of invariance is needed. On the other hand, complete invariance (sphere voting) depends only on the distance of the feature to the center and, as the result, the important spatial information is lost. We have observed that circle voting, which depends on the estimate of the one rotation invariant vector, which can be found robustly, gives the best performance. However, this outperforming method, circle voting, is the slowest method.

# Chapter 5

# Verification and expansion to improve shape search, classification, matching, and text-based shape search

This chapter joins previously introduced descriptions of meshes using 3D SURF (§3.2) with 3D ISM (§3.3) for 3D shape search. The 3D ISM introduces a weak (but fast) shape model that is crucial for 3D verification and query expansion which are the key parts of the proposed retrieval method. The system has no prior models of any object class and is class-generic. The pipeline that we propose is meant to put minimum efforts on the side of the user. To that end, it combines text-based and geometry-based search. This 3D shape retrieval is described in §5.3. It consists of two stages: i) a fast, yet somewhat sloppy verification, and ii) a query expansion step, where verified shapes are taken into account.

As an additional novelty, we demonstrate the usefulness of query expansion on shape classification with limited training data in §5.4.3 and shape matching in §5.4.4. The proposed 3D retrieval pipeline is also used to improve results of text-based shape search methods, such as Google Warehouse that tend to be quite incorrect. A shape-based consistency check then automatically reorders those results, without the need for further human supervision. We evaluate our shape search pipeline on a variety of datasets and, in particular, the combination with the text-based search is tested on a subset of a Google Warehouse of about

2.5K shapes. The method is described and evaluated in §5.5.

In the last part of the chapter (§5.6), we describe a real-life application of this retrieval algorithm, the 3D Coform shape search tool. The tool was used for the automatic geometry-based retrieval of archaeological artifacts in a museum's repositories. Finally, §5.7 concludes the chapter. The chapter was published in [84].

## 5.1 Introduction

The amount of available 3D data is rapidly growing thanks to portals such as Google Warehouse [62], as well as capturing systems such as Arc3D [186] or Photosynth [173]. As a result, the need for effective searching ([51, 181, 132, 85]) in these datasets is on the rise too. 3D shape retrieval is the problem of finding similar shapes - typically of the same object class - given a query.

Our approach extends the BoW baseline with two steps, which each boost performance. First, we verify the spatial configuration of the visual words for the shapes ranking high in the BoW search. Second, the shapes supported by this verification are used to expand the original query. This entire 3D pipeline, except for the BoW initialization, is novel for a 3D retrieval. The whole process is fully-automated, unsupervised, and class-generic. We explore these principles for their use with shape class detection and matching as well, and demonstrate that these applications do also benefit. Using these methods for such tasks is novel too.

In addition, most previous works aim at retrieving similar shapes when the query is also a 3D shape [165, 131, 21, 84]. Recently, a diversification in terms of the query type can be observed though. For instance, it can be given as an image [45] or a deformation of a base template through deformation controls [132]. The goals can also be set wider than sheer shape similarity, e.g. one can search for a shape that best fits into a given scene context [50]. As a matter of fact, the state-of-the-art leaves it quite open as to what would probably constitute the most often used retrieval mode: starting a simple text query. Indeed, it is not always realistic to assume that a user has a similar 3D shape model at hand, from which to kick off the search. Text-based search is already supported by Google-Warehouse [62] (GW), but as in the case of images, the query results are sometimes not what was hoped for in terms of real content.

Google Warehouse retrieves a sorted results list of shapes based on the similarity of the shape's text labels to the text of the query. As mentioned, the rank-ordered list of retrieved shapes tends to be quite noisy. We boost the quality of

the results in a fully unsupervised way, i.e. without requiring any further input or technical skills from the user. The central goal is to re-order the list, pulling the most relevant results up. So, if for example a query for "bike" returns a Star Wars space-ship at second rank, this should get pushed down, at the benefit of bike models, that ought to be ranked higher. This challenge is similar in flavor to Multiple Instance Learning [56], yet re-ranking of weakly labeled samples is the goal, rather than class discovery from scratch. We assume that the shapes of the actual target class form the largest group sharing multiple geometric features. From practical experience this tends to be the case when as few as 20% of the retrieved shapes are members of the target class. Shape similarity assessment will thus play a pivotal role in our approach.

The assessment of shape similarity is crucial in this scenario of re-ranking text-based shape search results. In our particular implementation, a two stage procedure is followed. It starts from a Bag-of-Words (BoW) approach [181, 132, 171]. The BoW method finds distinctive features on shapes, and matches them against a vocabulary of 'visual words' (quantized local 3D feature descriptors). The shape is then represented as a histogram of visual word occurrences. The similarity between shapes is measured in terms of the distances between the normalized versions of their BoW vectors, as in §3.4.2. . Such tests still only yield a weak guarantee that the shapes are really similar. That is why our pipeline also includes a second, more stringent stage. That stage goes after the layout information, i.e. whether the features are in consistent relative positions or not (rather than just considering their individual presence). Note that the relative configuration between neighboring features is different assumption than the ISM-like 3.3 star model that would weaken here because it would not have enough shapes to learn intra-class variety. For example, if the pose of two lions would differ (fig. 1.1), their heads will be in the different configuration to the object center. But head's configuration to a close patches (i.e. the neck) are still correct. Finally, the second stage (after BoW ranking) itself involves two steps. First, we verify the spatial configuration of those corresponding features that are ranked high in the list after BoW re-ranking. Second, the shapes supported by this layout verification are used to expand the original query.

## 5.2 Related work

We now discuss a variety of research directions our work builds upon.

**Local features.** Except for a few cases, such as the seminal spin images [74], it is only recently that 2D concepts like local features have made their entrance

as a mainstream approach in 3D searches [21, 181, 85, 170]. With them also came approaches for retrieval that earlier were used for images, like Bag-of-Words (BoW). The 3D versions of local features [85, 177, 170] have since then been combined successfully with BoW based methods [21, 181]. These approaches were shown to be robust to noise and orientation changes, and even to deformations. We also follow a local feature-based approach (3D SURF); however, the particular choice of feature type is not crucial to the proposed method. Our method starts off with BoW as well, but subsequently improves the shape retrieval with an important layout verification and query expansion step. As a matter of fact, these steps form a fully automated relevance feedback loop.

**Cascaded approaches.** Computational time and accuracy often have to be traded off against each other, when examining the semantic relevance of a search. Regardless of the chosen representation, most methods end up improving the search relevance in a somewhat cascaded or iterative manner. Assume that two different representations (feature or distance-wise) are constructed for an object: one that allows for a fast search and another that allows for the improvement of accuracy. The idea is to first quickly prune the set of candidates, to then spend more time on the promising candidates that are left by the first step. Accuracy can be improved in a variety of ways:

(i) Incorporating user feedback (a strategy that is still the most popular in 3D shape search [120, 46, 135, 68]).

(ii) Via structural consistency. This improvement is the most popular in 2D [143, 118, 32] and also exploited in our work

(iii) Through a variety of heuristics e.g. pseudo-relevance feedback, multiple queries [107, 10, 135].

(iv) Query expansion [32] that carefully expands the query object based on the relevant information from the dataset. We left query expansion as a separate point because it is the best improvement [118, 32] of BoW search algorithms and it was found to excel when searching is defined as a classification task [44] too.

With its verification and expansion steps, our method also forms a cascade, with novelties proposed for each step.

**Structural verification.** The BoW representations are compact shape models that typical come with a good level of scale and orientation invariance. They

capture the statistics of feature occurrences, but not of higher-order co-occurrences, let alone of configurational aspects. Word co-occurrences have been used in image retrieval [141, 73], as (more recently) have structural constraints [32, 7]. One can go further, of course, and perform full graph matching [133, 79], but such approaches tend to be too computationally expensive to be applied to even mildly-scaled retrieval. Therefore, we stop at considering such approaches, and stick to simpler layout verifications between matching pairs of local features.

**Combining textual and visual features.** Letting text-based retrieval be followed by visual refinement has mostly been used for image search, e.g. Luo *et al.* [109] built image clusters based on their visual features and then returned images falling in the same cluster as the query. Cui *et al.* [37] let users select one image among those returned by the initial, text-based search. Similarity is then redefined based on its visual features. Min *et al.* [119] also mixed textual and visual features. Their approach differs from ours in that, initially, the text- and shape-based searches are carried out independently, yielding two result lists. Finally, the two lists are integrated through an adapted weighting of both. The approach proposed here rather prefers shapes that have high within text-query similarity (fig. 5.6).

## 5.3 Shape search using verification

In this section we focus on how we measure the similarity between shapes in the different steps of our pipeline. As already suggested in the previous section, the level of sophistication of the measures is systematically increased, starting from a rather crude definition of similarity used for BoW (§2.1.4), and then a more refined one to take into account the relative spatial arrangements of matching local features (§5.3.2) that is performed on the subset of results of the first stage. This section also describes the similarity used for the last step, i.e. the expansion (§5.3.3) that is performed on the set of shapes that were successfully verified by the second step.

### 5.3.1 Initial Bag-of-Words search

Given a query shape, the goal is to quickly weed out the majority of unrelated shapes in the database. To that end, we use BoW, an approach that can be considered a good baseline in retrieval [171, 21] due to its robustness against noise and highly compressed shape representations that allow for efficient querying. In order to represent a 3D shape, local features are computed

according to generic criteria, such as saliency, repeatability, robustness, and invariance [74, 177, 85]. Features extracted from a training set of shapes are clustered to form "visual words" such as in §3.3.1. The object is then represented as a normalized histogram of visual word occurrences, also know as a bag-of-visual words. The similarity of objects is then measured as a distance between such BoW histograms, and shapes are sorted according to that similarity, which forms a results list.

### 5.3.2 Structural verification step

The BoW-based similarities are still pretty weak in only bringing up candidates that are truly relevant for a query [141, 82, 33]. Going after the co-occurrence of visual 3D words rather than occurrence, (as with BoW pure, would already be more strict as a test [21]. Yet, light-weight structural verification tests that check on the spatial layout of the visual words is more effective [141, 85]. By light-weight we mean some test that is substantially cheaper than a full graph-based comparison. Comprehensive object structure verification is computationally expensive, even when using state of the art simplifications [49, 133, 79] and would only be suited for very small vocabularies [21] with limited discriminant power.

For our tests we return to the individual feature descriptors (instead of the BoW histogram) of shapes. In particular, we consider the query shape $q$ and the list of candidate shapes $r$ coming out of the BoW search step in §5.3.1. Let $\mathcal{Q}$ and $\mathcal{R}$ be the sets of their features, respectively. We want to add tests on the different $r$ to check whether they are really relevant for $q$. In order to measure the difference in shape pairs, we use a symmetric Modified Hausdorff Distance, introduced by Dubuisson and Jain [42]. Distance between two shapes is then computed from the similarity of pairs of features between these shapes. Formally,

$$d_{MHD}(q,r) \;=\; \frac{1}{|\mathcal{Q}|}\sum_{\mathbf{q}\in\mathcal{Q}} \min_{\mathbf{r}\in\mathcal{R}}\big(\mathrm{dist}(\mathbf{q},\mathbf{r})\big) \;+\; \frac{1}{|\mathcal{R}|}\sum_{\mathbf{r}\in\mathcal{R}} \min_{\mathbf{q}\in\mathcal{Q}}\big(\mathrm{dist}(\mathbf{r},\mathbf{q})\big), \quad (5.1)$$

where $\mathbf{q}$ and $\mathbf{r}$ are some features in $\mathcal{Q}$ and $\mathcal{R}$, respectively This sums the distance of every feature from the set $\mathcal{Q}$ to the most similar feature from the second set $\mathcal{R}$ and vice versa, with $|\mathcal{Q}|$ the number of features in $q$. The distance $\mathrm{dist}(\mathbf{q},\mathbf{r})$ expresses our belief that feature $\mathbf{q}$ is a correct match for $\mathbf{r}$ and takes the feature layout in account. Its particular nature is explained in the next section. In practice, for example for the feature $\mathbf{q}$ in the first sum in eq. 5.1, we found that it suffices to search among the four most closest features from $\mathcal{R}$ in the descriptor space. This improves time efficiency significantly. We observed that

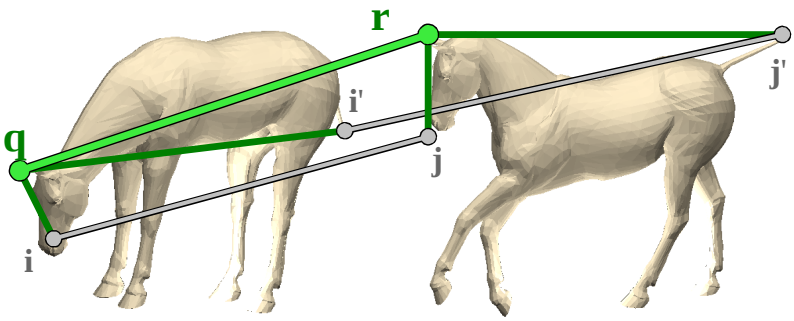



Figure 5.1: **Illustration of the verification method.** Given two shapes and a pair of features $\{\mathbf{q}, \mathbf{r}\}$, the distance between them dist$(\mathbf{q}, \mathbf{r})$ signifies if they represent a good correspondence. Instead of computing dist$(\mathbf{q}, \mathbf{r})$ from descriptors, we measure the consistency of correspondences in a neighborhood. This assumes that if $\{\mathbf{q}, \mathbf{r}\}$ is a correct match, $\{\mathbf{q}, \mathbf{r}\}$ will be surrounded by correct matches. For example, the candidates $\{\mathbf{i}, \mathbf{j}\}$ are in correspondence. Correspondences that are far (i.e. $\{\mathbf{i}', \mathbf{j}'\}$) are down-weighted to not be taken into account. This is an illustration, each $\{\mathbf{q}, \mathbf{r}\}$ is compared to hundreds of correspondences.

while considering less than four closest features leads to a loss in performance, four and beyond yields the same results.

### N part weak shape model (WN)

We want to define dist$(\mathbf{q}, \mathbf{r})$ in a way that preserves robustness by considering more than a single pair of matching features, such as point-to-anchor ISM model [85]. As the ISM learns intra-class variability from the training data, it will not be useful for this example-to-example similarity measure. Hence, in the vicinity of both features of a matching pair (a vicinity of each on each of the two shapes under comparison) we look for other features, that also match between the shapes. One can consider this using the matching features from which we started as centers. Yet, as the shapes deform, what we hope is that the vicinity will move with the features. As long as this assumption does not fail too often, our strategy will be able to handle the situation.

First, a set $\mathcal{M}$ of pairs of corresponding features between $\mathbf{q}$ and $\mathbf{r}$ based on the similarity of their descriptors is constructed. For every feature on the first shape, we find the set of four most similar features on the second shape, and vice versa The resulting pairs of candidate matches are collected in the set $\mathcal{M}$. To arrive at these pairs, we measured the similarity between the original feature descriptors

without using the BoW's quantized visual words. This came out to work better than reliance on the visual words. In order to reduce the computational cost behind getting $\mathcal{M}$, we used an approximate nearest neighbor search [121]. Thus, after this operation $\mathcal{M}$ stores pairs of possibly corresponding features, such as $\{\mathbf{i}, \mathbf{j}\}$.

Second, the similarity of the correspondence $\text{dist}(\mathbf{q}, \mathbf{r})$ now depends on the matching quality of other correspondences that are in the vicinity of $\mathbf{q}$ and $\mathbf{r}$. It means that the correspondence is correct if neighborhood correspondences are correct too (fig. 5.1) Formally,

$$\text{dist}(\mathbf{q}, \mathbf{r}) \;=\; 1 - \frac{1}{M} \sum_{\{\mathbf{i},\mathbf{j}\} \in \mathcal{M}} \exp\Big(-\max(\overline{\mathbf{qi}}, \overline{\mathbf{rj}})^2\Big) \exp\Big(-\big(\overline{\mathbf{qi}} - \overline{\mathbf{rj}}\big)^2\Big), \quad (5.2)$$

where the *sum* term measures the configuration of $\{\mathbf{q}, \mathbf{r}\}$ to every correspondence from the set $\mathcal{M}$. Let us see two parts of the *sum* on the example of how a $\mathbf{i}, \mathbf{j}$ affects $\mathbf{q}, \mathbf{r}$: the first part of the *sum* weights the $\{\mathbf{i}, \mathbf{j}\}$ correspondence high, if it is close enough to affect $\{\mathbf{q}, \mathbf{r}\}$. The second part weights $\{\mathbf{i}, \mathbf{j}\}$ high if it is in the correct configuration to $\{\mathbf{q}, \mathbf{r}\}$. Thus, the distance $\text{dist}(\mathbf{q}, \mathbf{r})$ will be low if correspondences that are close are in the correct configuration (fig. 5.1. $\overline{\mathbf{qi}}$ is the spatial asymmetric measure of the difference between two features $\mathbf{q}$ and $\mathbf{i}$ on the same shape and defined as: $\overline{\mathbf{qi}} = \|\mathbf{p}_q - \mathbf{p}_i\|/\sigma_q$, where $\mathbf{p}_q$ refers to the position of feature $\mathbf{q}$, $\mathbf{p}_i$ refers to the position of feature $\mathbf{i}$ and $\sigma_q$ refers to $\mathbf{q}$'s scale directly given from the 3D SURF feature [85]. The scale defines a vicinity of the feature $\mathbf{q}$ on which we work. Note that $\overline{\mathbf{qi}}$ is the asymmetric distance, where we always fix the scale for $\mathbf{q}$ or $\mathbf{r}$. The finale distance between two shapes $d_{MHD}(\cdot)$ still preserves the symmetry.

### 5.3.3 Query expansion

We now present the scheme to re-issue a new query using the relevant (and irrelevant) subsets that result from the verification procedure, for a reliable, augmented results list.

#### Average expansion (AE)

The most popular strategy to re-issue a new query is called average query expansion [32, 150]. The mean of the BoW vectors associated with the top ten shapes from the *verified, retrieved shapes* is used to construct a new query. For reliable verification, the expansion threshold from the first set of verified

results must be set carefully to avoid inclusion of incorrect documents, as it will destroy the query BoW vector (the validation set is usually used here, we tuned the threshold for one dataset that has validation data, then the threshold was kept for the rest of experiments).

#### Average expansion with negatives (AEneg)

We want to employ information from both the positively and the negatively verified samples. Given a BoW vector $\mathbf{b}_q$ of the query shape and the result of the verification stage, the goal is to estimate a new BoW query vector ($\mathbf{b}'$) that will support positively verified shapes (as in AE), and lower the similarity of $\mathbf{b}'$ to the negatively verified shapes, as these negatively verified shapes were incorrectly similar to the query. To achieve this, in addition to augmenting with the mean of the positive examples (similar to the original AE described above), we decrease the new BoW vector by the mean of BoWs of shapes that were verified to be negative,

$$\mathbf{b}' = f\left(\frac{1}{P+1}(\mathbf{b}_q + \sum_{\mathbf{b}\in\mathcal{P}} \mathbf{b}) - \frac{1}{N}\sum_{\mathbf{b}\in\mathcal{N}} \mathbf{b}\right) \tag{5.3}$$

$$f(\mathbf{k}) = \begin{cases} \mathbf{k} & \text{if } \mathbf{k} > 0 \\ 0 & \text{if } \mathbf{k} \leq 0 \end{cases} \tag{5.4}$$

where $\mathcal{P}$ is a set of BoWs of positive results shapes, $\mathcal{N}$ are the negatives and $\mathbf{b}_q$ is BoW of the query shape. The $f(\cdot)$ function avoids negative values in BoW where the effect of negative samples is bigger than the positives. Note that normalization is included later in the tf-idf weighting of $\mathbf{b}'$.

## 5.4 Evaluation of verification and expansion

Shape search algorithms are evaluated here. We firstly introduce several competitors in §5.4.1. Then, §5.4.2 evaluates verification and expansion on the task of shape search and §5.4.3 on classification.

### 5.4.1 Competitors

We first introduce three popular shape enhancement competitors for our approach. The subsection describes three methods of initial search improvement and two modifications of the proposed WN.

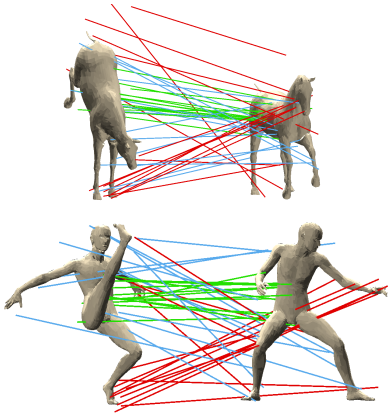


Figure 5.2: **Features that were selected for DMLL training.** Example of pairs of shapes that were verified to be relevant to each other. Three groups of features are highlighted: i) **red** for negatives, where features are matches after nn-matching but they have high WN distance; ii) **green** for positives, which have low WN distance; iii) **blue** shows the random set.

### Latent Dirichlet allocation (LDA)

The LDA approach is a popular method for text and image search where a document is represented by a mixture of topics (see more in [143]), and every topic is a distribution of words. We run standard a LDA [143] on all our datasets as a competitor.

### $K$-reciprocal nearest neighbors (KRN)

This is a simple, yet effective method [148] for shape search reordering where initial search results are used to define a different nearest neighbor metric based purely on symmetry of mutual BoW similarities. In this method, the $i$-th result shape ($s_i$) in a query's retrieved list is considered verified if the query appeared in the first $K$ results of the query using $s_i$. For this method, we ran experiments for different $K$ and selected the one with the highest performance (see Qin *et al.* [148]). The specialty of this method is its speed and ability to use any (even global) descriptor.

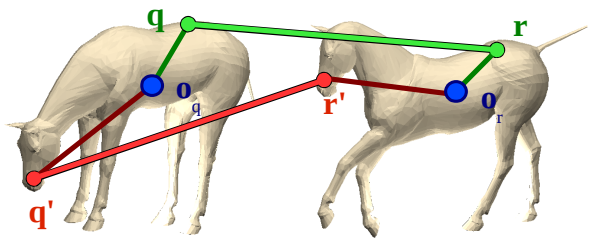



Figure 5.3: **W2.** We also experimented with using the ISM model in the verification stage. In W2, Each match of features is weighted by its relation to the object center instead of the configuration to its local neighbors as in Wb.

### Distance metric learning for large margin (DMLLM)

We used Philbin *et al.* [144]'s approach for unsupervised 3D shape distance learning. After a initial search and WN verification, we performed feature matching which becomes these three sets of features:

(i) Correct (high WN similarity, which is **green** in fig. 5.2).

(ii) Not correct (**red** in fig. 5.2).

(iii) Features of random negative pairs to decrease the confusion between negative and positive training features (**blue** in fig. 5.2).

Then, we follow the work of Philbin *et al.* [144] and we learn a mapping of descriptors to separate these three sets from each other. When the mapping is learn, we project every shape's feature descriptor on to the new space and the shape search pipeline was recomputed to obtain new results.

### W2

This is a *two part weak shape model* modification of WNwith the most simple spatial layout á la ISM [104, 85]. The method assumes that the observation of any one feature (part 1, feature) in a specific configuration w.r.t. to the object center (part 2, anchor) is sufficient to verify the shape. We use this observation to reformulate dist$(\mathbf{f}, \mathbf{g})$ in eq. 5.1. The distance is low when the relative distances of $\mathbf{f}$ and $\mathbf{g}$ to the shape's centers are similar. Formally,

$$\text{dist}(\mathbf{f}, \mathbf{g}) = \left(\overline{\mathbf{o}_q\mathbf{f}} - \overline{\mathbf{o}_r\mathbf{g}}\right)^2 / \sigma_{W2}^2, \tag{5.5}$$

| | *TOSCA* | | | | *Princeton* | | | | *SHREC'09* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| method | verifi. | AE | AEneg | PC | verifi. | AE | AEneg | PC | verifi. | AE | AEneg | PC |
| W2 | 0.624 | 0.635 | 0.649 | 0.634 | 0.283 | 0.337 | 0.339 | 0.326 | 0.262 | 0.366 | 0.368 | 0.290 |
| WN-global | 0.635 | 0.713 | **0.718** | 0.671 | 0.284 | 0.335 | 0.332 | 0.321 | 0.267 | 0.389 | 0.389 | 0.296 |
| WN | 0.626 | 0.647 | 0.663 | 0.672 | 0.283 | 0.334 | 0.336 | 0.323 | 0.285 | 0.423 | **0.425** | 0.313 |
| KRN [148] | 0.626 | 0.667 | 0.684 | 0.668 | 0.283 | 0.322 | 0.322 | 0.308 | 0.275 | 0.338 | 0.340 | 0.283 |
| LDA[143] | 0.627 | | | | **0.344** | | | | 0.328 | | | |
| DMLLM[144] | 0.632 | | | | 0.272 | | | | 0.262 | | | |
| Initial BoW[85] | 0.622 | | | | 0.282 | | | | 0.277 | | | |

Table 5.1: **Shape search results**. Performance is measured as the average area under the PR curve. The proposed verification and expansion outperform the basic search. While incorporating negatives into the average expansion improves the performance, it is not very significant. The WN-global and W2 are variants of the proposed method, and they are discussed in the text.

| data/method | Initial BoW[85] | WN | WN+AE | WN+DMLLM | LDA[143] |
|---|---|---|---|---|---|
| *TOSCA low res.* | 0.503 | 0.552 | **0.571** | 0.520 | 0.427 |
| *Princeton* | 0.283 | 0.284 | **0.293** | 0.280 | 0.241 |
| *SHREC'09* | 0.145 | 0.143 | **0.183** | 0.149 | 0.101 |

Table 5.2: **Shape search results**. Performance of the winners from tab.5.1 on small vocabulary (50 visual words). This allows the use of learning the DMLLM method [144] as a method for expansion, where a new metric is learnt from verified and not verified shapes. Interestingly, the performance of LDA dropped significantly.

where $\mathbf{o}$ is the shape center and $\sigma_{W2}$ corresponds to 1% of the unit shape size and it controls the maximal error between matches $\mathbf{f}$ and $\mathbf{g}$. It is shown in fig. 5.3.

#### WN-global

This modification of WNuses a global point scale. Herein, $\sigma_q$ corresponds to the size of the shape. So that, the parameter is estimated globally, and it is constant for all shapes' points.

### 5.4.2 Evaluation of shape retrieval

Having introduced the competitors in §5.4.1, we pitch combinations of our verification and expansion against them. We test this on the established datasets: *Tosca* [20] of 120 hand-made shapes of 9 classes, *Princeton* [166] of $1.8K$ shapes and *SHREC'09* [43] with 20 challenging partial queries for 720 shapes. We also run our methods on *SHREC'10* [21], where results of the initial search are visually similar to BoW of Bronstein *et al.* [21], but the ground-truth to evaluate results was not sent to us.

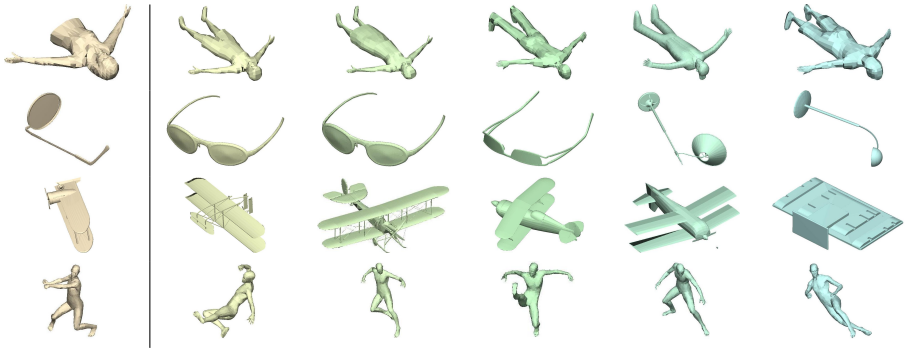



Figure 5.4: **Shape search results.** Every row starts with a query shape and then shapes sorted by rank continues. Note that query expansion adds diversity to the results list, as shapes have to be similar to all shapes that were used for expansion.

Results are shown in tab. 5.1 and fig. 5.4. We can conclude that verification with expansion significantly improves initial results. The LDA gives significant improvement (and is even the winner for the *Princeton* dataset), while it has no spatial information. On the other hand, LDA does not perform well on *TOSCA* and has significantly decreased performance when small vocabularies are used (tab. 5.2). The WN-global gives retrieval improvement on complete, clean shapes (*Tosca*), while local information matters when only partial queries are given, such as in *SHREC'09*. Note that this method is still local and only the parameter for the feature's neighborhood is estimated globally. While W2 improves the initial search, the performance is mostly bellow the performance of the proposed WN. In conclusion, we note that local verification is important. A number of shape search results after expansion are presented in fig. 5.4. Though performance can depend on the specific dataset, verification with expansion generally leads to improved shape search. This demonstrates the utility of incorporating spatial layout and using class-specific information (that could be missing in the original query).

We have also observed a small improvement when incorporating negative examples for the expansion (AE vs. AEneg in tab. 5.1). To increase the number of expansion methods, we also included Pairwise coupling (PC) [190] (implemented in libsvm [28]). The method learns a score-based classifier between the two sets of features (positive and negative sets are formed as in AEneg). However it also improves the verification stage, its improvement is comparable with WN/W2/WN-global on the TOSCA dataset only. It worsen on other datasets.

| | *TOSCA* | | *SHREC'09* | |
|---|---|---|---|---|
| method | knn | SVM | knn | SVM |
| ST: standard | 89.4% | 79.0% | 50.0% | 60.0% |
| **OE: one-example** | 47.3% | 54.6% | 45.0% | 40.0% |
| **OE+QE: one-example+QE** | 63.2% | 63.2% | 45.0% | 65.0% |

Table 5.3: **Performance of the classification**.

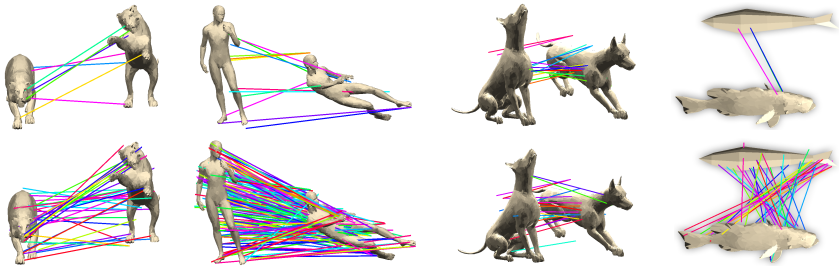


Figure 5.5: **Matching.** First row shows results of matching with original features, second shows the same matching algorithm and parameters on original features plus expanded.

### 5.4.3 Expansion for classification

Here, we present expansion and verification to improve classification when only a few training examples are available. In tab. 5.3 results are shown for three types of classification:

(i) Standard classification (ST) that is learn on a large amount of labeled training data.

(ii) One-example classification (OE) that uses only one per-class randomly selected shape for training. As expected, the performance of OE drops considerably.

(iii) One-example classification and query expansion (OE+QE). For training, we use only one example per class again. In addition, we used the rest of training data for expansion of the training set as well as the test set.

This experiment uses the same datasets as in §5.4.2. To show that our method works independently on the choice of the classification method, we perform experiments on two types of classifiers: SVM [181] classifier and the k-nearest neighbor [17] classifier. Tab. 5.3 shows that standard classification is the best one, because it uses complete set of the training shapes. One-example

classification is, as expected, the worst one and it is improved by expansion. The OE+QE even outperformed ST in one case, (probably because the queries, often partial shapes from SHREC'09, benefit from the expansion exploiting relevant structure).

### 5.4.4 Expansion for denser shape matching

Here we present an example of using expansion to improve shape matching. Figure 5.5 shows results of matching shapes with original and expanded features, as described in §5.3.2. This attempt at matching, ignores the more sophisticated conditions of matching (as in [147, 80, 133], but estimates correspondences in less than $0.5s$ (compared to tens of minutes for [147, 80, 133] or minutes in Hungarian). But this initial attempt seems promising for a deeper application in this field. As the idea is to add new relevant features from the database, matches are more denser when exactly the same algorithm is used.

## 5.5 Text-based shape search

The proposed approach for shape verification and expansion was already discussed in §5.3. We now introduce text-based shape search and will present the application of our 3D shape search pipeline to the text-based shape search. The experimental part starts with the introduction of datasets in §5.5.1. Then, we test and discuss the proposed method in §5.5.3.

Text-queries often return a large number of wrong results, possibly due to their heavy relevance on only meta-data for retrieval (e.g. among the top results for a "bike" is a Star Wars spaceship). We want to employ actual 3D data for improved retrieval and the extension of our efforts on verification and expansion here. Additionally, the primary means of retrieval here are text based queries employed by professionals (e.g. animation). The goal of text-based shape search it retrieving 3D shapes that corresponds to the text-query.

Text-based shape search idea.

### 5.5.1 Data

We start with a short introduction of the dataset. It naturally introduces notations that are required later.

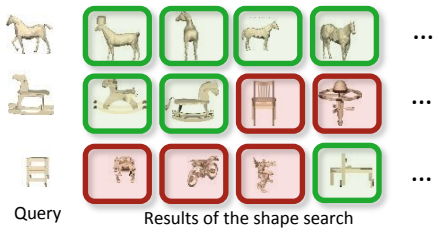



Figure 5.6: **Text-based shape search idea.** Top GW text-query "horse" shapes are shown at the most left column. For each, we sort **whole** database based on the 3D similarity. Green color highlights if the shape from the database has same query-text tag. ("horse" in this case). Shapes that truly corresponds to the "horse" have majority of relevant shapes in the top.

Given a text-query string, the text-based 3D search engine retrieves a set of shapes that are sorted according to the similarity of the text-query and text meta-data [62] associated with the shape. The result of this crawling process defines our *database* of shapes where every shape, $q$, is stored with: i) a text-query tag $t_q$ for which it was retrieved and ii) the rank in the results list, which corresponds to the text-query, in other words, how the shape is relevant to the text-query.

We have downloaded the 150 best results for several text-queries, such as "bike", "motorbike", "cat", "table", "chair", "car" etc. accumulating a total of 2.5K shapes. For every shape we store only its rank in the Google-Warehouse's (GW) result list and the query-tag (often incorrect) of the shape. We work with automatically approximate ground truth as labeling huge datasets is often time consuming. The process is as follows, the user labels about 15 correct shapes for every query-tag. Then, a one-against-many SVM is learnt to classify the rest of the database. This labeling was used as *one* approximation of the ground-truth, to evaluate shape ordering.

### 5.5.2 Improving text-based shape search

We describe a method to improve text-based shape search using geometrical verification and shape search. This method processes a dataset offline. It encourages shapes with high within-class similarity and discourages shapes with high other-class similarity. The method works in the following way.

For every shape, $q$, we performed our shape search method described in §5.3 on the complete database to compute a *query-Text-Relevance Score* ($trs$). Once we run the shape search on the complete database, class-relevant shapes that corresponds to query-tag of $q$ share a 3D content and will be at the beginning of the result list, as shown for "horse" in fig. 5.6. We use this geometry based ranking to estimate $trs$. We define the $trs$ as a function that favors shapes with same query-tag at the beginning in the result list. Thus, the $trs$ for shape $q$ is

**Input:** Database = a set of shapes. Each shape is associated with its rank and its text-query tag.
**Output:** Updated rank of shapes.
**for** *each shape* $q$ **do**
  ▶ Use §5.3 to sort whole database based on the similarity to $q$.
  ▶ Calculate $trs(q)$ using eq. 5.6.
**end**
**for** *each text-query* **do**
  ▶ re-sort shapes based on their trs score.
**end**

**Algorithm 1:** Text-based shape search improvement.

defined as the weighted portion of shapes for which shape-retrieval (using $q$ as a query) gives text-label consistent results. Formally,

$$\mathrm{trs}(q) = \sum_{i=1}^{D-1} \exp\left(\frac{-f(t_i = t_q)^2}{\sigma^2}\right), \tag{5.6}$$

where $t_q$ is the tag of shape $q$ and $t_i$ is the tag of the $i$-th most similar shape. Then $f(t_i = t_q)$ returns $i$ if the $i$-th shape was downloaded for the same text-query tag as shape $q$; zero is returned otherwise. The parameter $\sigma$ encourages the shapes at the beginning of the results list. Note that the shape search is the key element of the algorithm. For this shape search, we used our complete pipeline including the initial search, geometrical verification, and query expansion.

Once $trs$ is calculated for each shape, we **re-rank** all results for each query-tag based on $trs$. The algorithm is summarized in alg. 1.

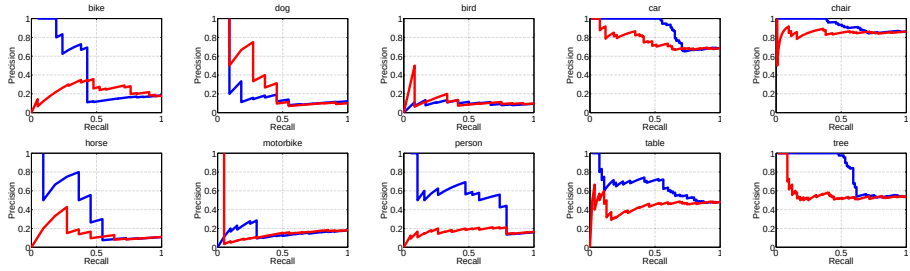



Figure 5.7: **Performance on text-based shape search.** Blue Precision-Recall curves corresponds to our method, red to the result of pure text-based search of Google Warehouse [62].

### 5.5.3 Experiments

We now evaluate our proposal for text-based shape search improved by the geometry of the shape.

Results are shown in fig. 5.7. For the *trs* ranking, we observed that results are dramatically improved even if only a small number of correct shapes were downloaded for a given query-tag (e.g. for "bike" and "horse" 40% of the shapes corresponds to the query-tag). However, an extreme lack of correct shapes in classes such as "cat" or "bird", where the percentage of correct shapes is less than 10%, can produce poor results. Please note that the performance also depends on the shapes with different query-tag as these shapes avoid wrong query-tag shapes.

## 5.6 Shape search on archaeological data

The proposed method for shape search was also practically tested for retrieving archaeological data in the 3D Coform project. The goal of the shape search tool is to find relevant archaeological artifacts (tested on sculptures) based on geometry. A 3D representation was obtained using Arc3D [186] or a scanning device, methods that the project's partners focused on. We now describe how the retrieval pipeline, already described in this section, communicates with other tools.

**Problem statement.** Figure 5.8 overviews the integration of the shape search tool. Tools that are outside KUL are on the left side, while programs that run on the KUL server are on the right side. The communication was implemented via web-service. The data flow is as follows:

(i) Feature computation: First, we compute 3D SURF features of the 3D model on the side of the archaeological institution. The place where the feature calculator runs is a strong requirement as a 3D model of the artifact cannot leave the institution due to the copy-right issues. Once the 3D SURFs are calculated, the calculator sends them to KUL.

(ii) The KUL server is constantly waiting for new requests while in sleep mode. When data was sent by the calculator, the KUL server indexes it into a visual vocabulary.

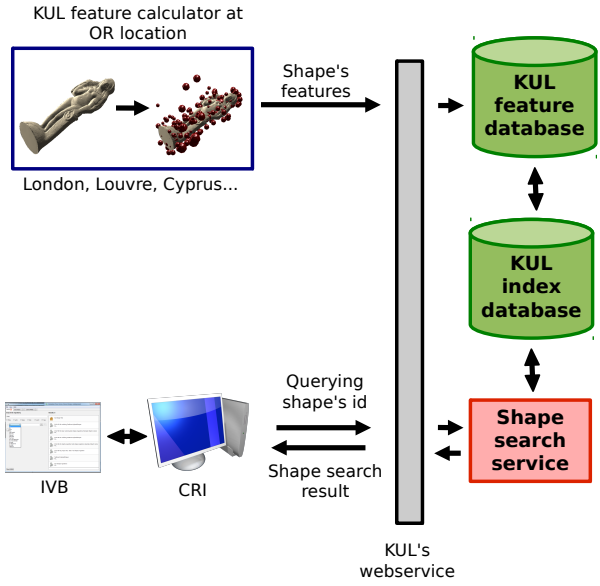



Figure 5.8: **Communication of the shape search tool's components.** Features are computed at the OR location. Then, only features are sent to the KUL server where visual words are calculated and the shape's indicates stored. After this is done, the shape appears in the results list and can also be used as a query. Once a significant amount of shapes are added, the visual vocabulary is recomputed and all indicates recalculated.

(iii) When the indexing is done, a new shape can be used for querying via the CRI (communication interface) and also appears in the results list in the visualizer (IVB).

(iv) If a significant number of new shapes have been added to a database, the visual vocabulary does not describe the data well. Then, as an additional feature of the tool, we back-up the current state and we create a new visual vocabulary and re-index all the data. If any data was added during the process of creating the new visual vocabulary, the indexing is only postponed until the new visual vocabulary is ready.

**Provided services.** Our shape search tool provides these services:

(i) Shape search: sort shapes in the database by similarity given an id of the query shape; the communication is done via web-service.

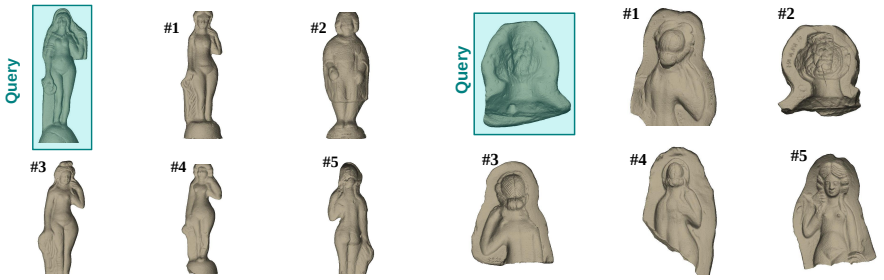
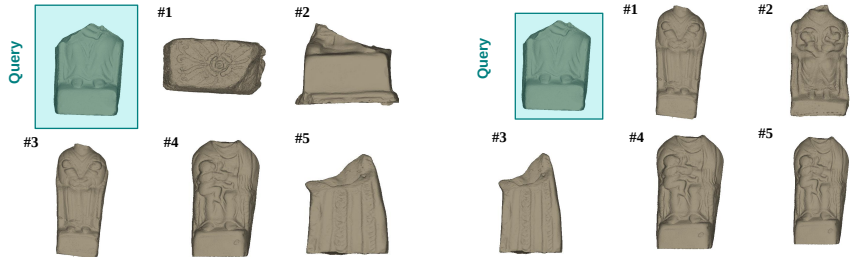



Figure 5.9: **3D Coform shape search results.** Two examples where the initial BoW shape search performs well.




(a) BoW result (b) Spatial verification result

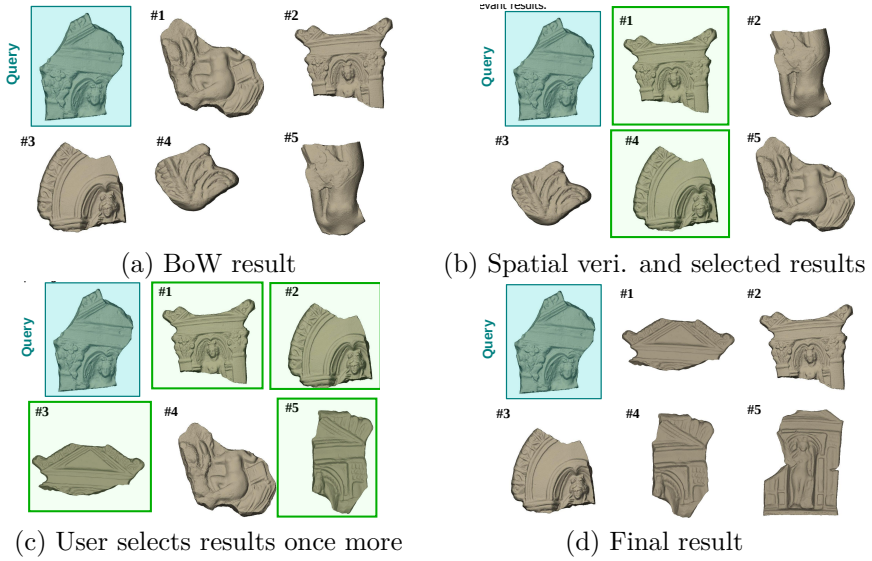
Figure 5.10: **3D Coform shape search results.** The initial shape search does not perform well (a), but the verification step improves the performance (b).

(ii) Visualization: an html page can also return results by a rendering of shapes. For this, renderings have to be pre-computed for the database. When results are shown, the user can easily select a new query by simply clicking on the shape of his interest.

(iii) Features computation: This algorithm computes features for the shape at the OR location and the features (only features) are automatically sent to KUL for indexing into the database.

**Implementation details.** The KUL server communicates with partners by generating html pages using php and C++ scripts. The php script on the side of the KUL server side calls C++ code executing the shape search.

**Experiments.** In the 3D Coform project, we experimented with the set of 300 shapes from the Louvre museum. For our test purposes, we run the BoW




(a) BoW result (b) Spatial veri. and selected results

(c) User selects results once more (d) Final result

Figure 5.11: **3D Coform shape search results.** Initial shape search does not perform well (a) and the verification step does not improve results neither (b). In this case, specialists can iteratively mark result shapes (green) of their interest and the algorithm improves results using the query expansion method (b,c,d).

(§5.3.1) shape search with the spatial verification (§5.3.2). Query expansion was included as well, but query shapes were expanded by 3D shapes that were marked by the user. In this case, it is the user that decides which results are relevant and which are wrong. We show only qualitative results here as the dataset was without ground-truth and restricted for our indoor testing purposes only. In fig. 5.9 we show cases where initial BoW method works well. Fig. 5.10 presents results where the initial BoW does not perform well, but results are improved by the spatial verification step. In the last experiment, we show an example where the spatial verification step does not perform well. Thus, the user can mark shapes of his interest and the query expansion method improves results towards the need of the user (fig. 5.11).

## 5.7 Conclusion

As promised, we improve upon the state-of-the-art 3D shape retrieval with a simple, yet reliable, cascade method of verification and expansion, relying on

weak structural verification. This allows us to have significant improvement while using an automatic, unsupervised and fast scheme.

While primarily developed for retrieval, the generality of the presented method and the interplay between the retrieval and the other areas of visual learning, provides us with a rich playground of problems to improve upon. We demonstrated this by attempting the challenging task of categorization in the presence of extremely limited and noisy data and demonstrated marked improvement. Though 3D databases are expanding rapidly, a variety of their clientele (such as animators) are interested in some categories more than others, thus making the role of categorization in retrieval particularly important. We show dramatic improvement on the results of the search giant Google on a reasonable subset of their 3D Sketchup warehouse.

# Chapter 6

# Segmentation of 3D scenes

So far, we have introduced a method (Chapter 3) to classify segmented and background-free shapes. This remains an artificial problem as more realistic scenarios include processing real-data with missing parts, occlusion, and a lot of background clutter. Thus, we introduce a method for finding objects in a scene, recognizing them, and segmenting the object's parts from the background here and in the following Chapter 7.

The chapter was published in [83] and we start with the introduction and related work discussion in §6.1. The method is defined in the following §6.2. Then, the text continues in §6.3 where §6.3.1 defines the segmentation as the graph cut problem. §6.3.2 shows how to use the recognition technique for the segmentation and §6.3.3 defines the connectivity between vertices on the scene. The experimental setup is explained in §6.4 and we conclude in §6.5.

## 6.1 Introduction and Related work

3D reconstruction and scanning methods have matured in recent times. This enables us to process 3D data for further tasks. Systems like the Xbox Kinect [72] now have the ability to retrieve 3D scans of scenes. While they are used for the purpose of pose estimation, they also allow for more efficient methods of shape classification, recognition, retrieval, and general scene understanding. Typically range-scanning or Structure-from-Motion (SfM) systems do not differentiate between the individual objects in the scene in the process of reconstruction. A raw 3D point cloud is usually the output of these systems and it is then converted

to a mesh. The process of mesh construction is naïve and does not differentiate between connecting parts of the same object or splitting two different objects into separate meshes. There's also inherent noise and ambiguity in the point clouds, themselves. In this context, the process of combining the process of recognition and segmentation can be useful for the purpose of scene understanding and recovery. This can help us make more complex inference, for applications. Understanding relationships further helps in object recognition in the presence of noise. What's more, domain knowledge of the objects can be used even for complex tasks of object completion etc.

The goal of our work therefore is to find objects in cluttered 3D scenes. We leverage Implicit Shape Model (ISM)-type algorithms to do so. Subsequent scene segmentation yields an interesting method of meshing the relevant parts together, which is a useful by-product.

This work draws inspiration from the worlds of relatively recent research in 3D object categorization and retrieval, and the more established field of corresponding problems in 2D. In 2D a variety of works were proposed for 2D object detection ([187], [104], [95], [167]). Motivated by the problem of face detection in [187], Haar-like filters are learnt in a boosting framework to train classifiers for face detection.

2D image segmentation has also been a well researched field. Following the clustering based approach of normalized cuts [164], the Markov Random Field (MRF) based methods of MinCut [19, 153] became very popular. While the original application of MinCut had very simple neighborhood based consistency priors, subsequent versions applied sophisticated unaries and data-dependent pairwise terms, resulting in a Conditional Random Field (CRF) [153, 95]. The ability to easily inject higher-level object detection information in the context of class characteristics makes MinCut-based CRF segmentation appealing from both a theoretical and an implementational point of view.

In 3D, a vast exploration of feature detectors and descriptors [160, 74, 188, 85, 177] has allowed exploration of further tasks. While curvature and surface based similarity measures were used for object detection, invariance and occlusion can be handled far more easily with features. Adaptation of BoW-based methods, similar to text and image retrieval have inspired shape retrieval methods such as [131].

The same BoW features are used to train classifiers for class recognition in [181]. A more recent approach of Nishino and Bariya [13], treats the object recognition problem as one of finding scale-invariant correspondences between the model and the target scene. To make the problem tractable, matching is performed hierarchically to ensure that subsequent matches maintain geometric

consistency with the previous ones. This enables culling the space of potential possibilities and performing efficient matching for objects. The downside is that each object must have a tree representing it, and it is not clear how well this approach would generalize for variation due to deformation. Here, we are additionally interested in the ability to detect instances of an object class, in the presence of noise, clutter and deformation in a populated scene. In other words, we are interested in class-specific detection than the object-specific. A focused sub-problem in this area deals with specific problems like reconstructed cities and architecture (the labeling problem of [122, 23, 61]). Information about the ground plane or several distance constraints are employed for good performance. In this work, we try to learn and employ more general shape models (ISM) without this kind of specific information.

There has also been increasing interest in the area of 3D segmentation. Kalogerakis *et al.* [76] learn about segmentation of shapes and configurations and subsequently treat it as a CRF optimization problem of finding segments and their mutual configurations. Such a method can be used for prototyping, texturing etc. While they tackle the problem of learning seamlessly joined segments with functional roles on a single object's surface, we concentrate on learning to segment class-specific instances in scene environments. Also in their approach, the supervised learning involves giving segmented training data, as opposed to ours which can learn from shape instances alone.

We combine the top-down cues of object class detection with the bottom-up MRF-based segmentation techniques for combined recognition and segmentation. The goal is to label each scene vertex by each object (or background's) label. We combine object localization (using ISM [85, 104]), with Min-Cut based segmentation [19] in order to partition a mesh-based scene graph into its component objects.

We employ recognition techniques instead of hand-tuned applications (connections to background plane, user-marked segments etc.).

## 6.2 Problem statement

Given a scene of vertices and edges, our goal is to determine where the objects of the learned class $\ell$ occur if we have the corresponding previously learned model of the class $\ell$. The mesh representation of the scene gives us naturally a graph for this problem setup. To solve the problem of object detection and segmentation we combine the object localization technique of ISM with the object segmentation technique of Graph Cuts.

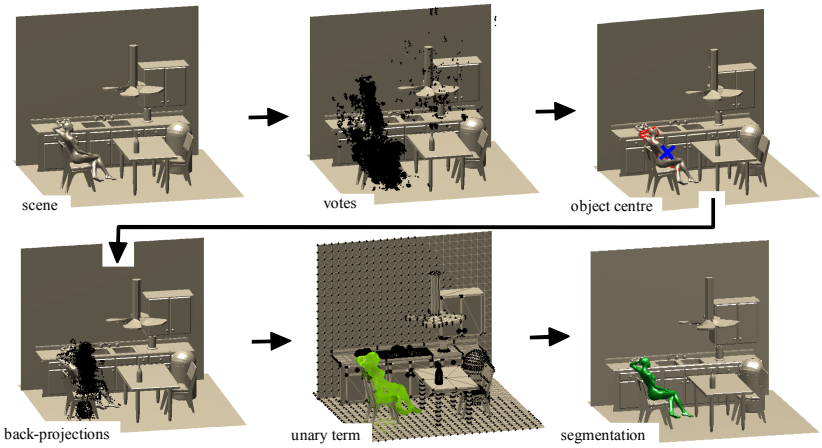



Figure 6.1: **Scene segmentation using the ISM based unary term.** Here, we were searching for the *person*-class. First, the scene of vertices and edges is given. Secondly, every vertex was assigned to the closest visual word based on its descriptor and then generated the set of votes on where the center of the object is located (black points). Third, the maximum density of votes defying the estimated centre of the class (blue) and we also have the contributing vertices (red points). Fourth, using the estimated centre, we back-projected all class points (black) which defines the unary term as shown by the green vertices in the fifth figure. Finally, MinCut gives the results shown in the last figure.

In the object localization phase, we learn the appearance of 3D object classes from training data. The learnt object class model can be used to classify and detect a new instance of the object class (as shown by [85, 181, 156]). However, in the absence of a clean object and in the presence of scene-based clutter and noise this is a challenging problem.

We use an ISM to detect and locate multiple object instances in scenes. The next step of object segmentation involves specifying an energy function and efficiently optimizing it. In our case, the energy function consists of two terms: i) a unary term defining the probability that the current graph node (scene vertex) belongs to the class of interest; ii) a pairwise term to model the probability of two neighboring nodes in the graph having the same label.

## 6.3 The method: ISM and MinCut for object segmentation

In this section, the proposed approach is discussed. In §6.3.1, we formulate the segmentation task as an optimization problem on the graph. §6.3.2 defines the unary term (using 3D ISM) and the section is concluded with the pairwise term in §6.3.3.

### 6.3.1 MinCut for efficient object segmentation

A scene defines the graph that is naturally given by the scene's mesh (its vertices and edges). Then, the scene segmentation problem can be posed as the labeling task, whether if each scene's vertex $\mathbf{v}_i$ belongs to class $\ell$ or not. The labeling is defined as a binary valued vector $\mathbf{x} \in \{0,1\}^{|\mathcal{V}|}$ with size of the number of vertices. So that, given the class-model $\mathcal{T}^\ell$ (ISM) and the connections between vertices (edges), the problem can be solved by minimizing the labeling function. Terms of the labeling function incorporates our goals of segmentation:

(i) Unary term takes into account a prior knowledge whether a point belongs to the foreground or the background.

(ii) Pairwise term takes into account that points that are next to each other should look similar and they should have the same label.

The above points are written formally as,

$$\mathcal{X}\big(\mathbf{x};\ \mathcal{V},\mathcal{E},\mathcal{T}^\ell\big) = \sum_{\mathbf{i}\in\mathcal{V}} \underbrace{x_i}_{\text{label}} \underbrace{\cdot\xi\big(\mathbf{i};\ \mathcal{E},\mathcal{T}^\ell\big)}_{\text{unary}} + \sum_{\{\mathbf{i},\mathbf{j}\}\in\mathcal{E}} \underbrace{\psi\big(\mathbf{i},\mathbf{j};\ \mathbf{v}\big)}_{\text{pairwise}}. \tag{6.1}$$

The first term $\xi(\cdot)$ is the **unary term** and it defines the negative log likelihood of the probability that the vertex $\mathbf{v}_i$ belongs to the class $\ell$. Please note that the background probability corresponds to $1-\xi(\cdot)$. The second one $\psi\big(\cdot\big)$ is called the **pairwise term** and it represents the negative likelihood of the probability that two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ on the edge $\mathbf{e}_l$ have the same label. In other words, the unary term models the probability if the vertex (graph's node) belongs to the class or the background and it is computed independently to other vertices. The pairwise term defines dependencies between vertices.

The solution of $\mathcal{X}(\cdot)$ can be efficiently found using the MinCut technique [19], which directly assign class labels to the vertices. The computation of the unary and pairwise terms are described in the following sections.

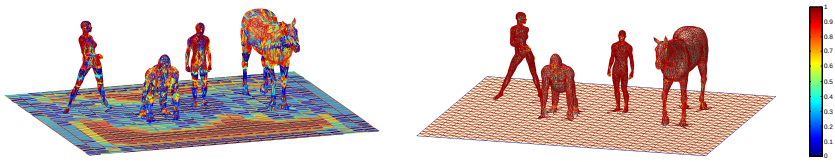
### 6.3.2 Definition of the recognition based unary term

With the success of Hough Transform based classification techniques [104, 85, 156], we propose the following way to define unary term. Using the ISM, we perform voting for the object's centre from every scene vertex, based on its closest visual word (fig. 6.1). The maximum density in the voting-space defines the estimated centre $\boldsymbol{\lambda}$ of the object. Using $\boldsymbol{\lambda}$, we back-project all votes from the class of interest. In other words, we generate the class-specific shape's structure as the following set of points, $\mathbf{b} = \{\mathbf{b}_j = -\mathbf{v}_j^* + \boldsymbol{\lambda} \mid j^* = 1...|\mathcal{T}^\ell|\}$, where $\mathbf{v}_j^*$ is a vote to the object's centre from the training data to class $\ell$, the training data to class $\ell$ have size $|\mathcal{T}^\ell|$, fig. 6.1 shows back-projections.

Thus from $\mathbf{b}$ we get a rough idea of where the object of the class might be and how it could look like. Now we compute the value of the unary term for every scene vertex as the weighted mean distance between the vertex and the $n$ closest back-projections,

$$\xi(\mathbf{i}\ ;\ \mathcal{E}, \mathcal{T}^\ell) = 1/n \sum_{k=1}^{n} \exp\left(\frac{-||\mathbf{p}_i - \mathbf{b}_k^{\mathbf{i}}||^2}{\sigma^2}\right) + f_l(\mathbf{i}), \tag{6.2}$$

where $\mathbf{b}_k^{\mathbf{i}}$ is the $k$-th closest projection for the $\mathbf{i}$ vertex, $\mathbf{p}_i$ is the 3D position of the $i$-th vertex $\mathbf{v}_i$. In our experiment we set $n$ to ten and the function $f_l(\mathbf{i})$ returns one if $\mathbf{i}$ correctly voted for the estimated $\boldsymbol{\lambda}$, otherwise zero. Parameter $f_l$ is a very strong condition as it needs the correct vertex's visual word in the correct configuration, this is not accomplished for all class vertices due to noise, occlusions and feature quantization.



(a) Vertices similarity (b) Plane-like surface

Figure 6.2: **Pairwise term.** Visualization of two different methods for the pairwise term. **Red** color represents edges with strong connectivity between vertices, which corresponds to $\psi(\cdot) = 1$. **Blue** shows edges with no connectivity, $\psi(\cdot) = 0$. Values of the color-bar correspond to the scale of the pairwise term $\psi(\cdot)$.

### 6.3.3 Definition of the pairwise term

This section presents three different definitions of the pairwise term. They are later compared in the experimental section.

**Pott's model.** For vertices which are directly connected by the edge, $\psi(\cdot)$ is identical one. Otherwise it is zero.

**Vertex similarity.** The descriptor vector determines that similar shape structures have similar surroundings and different structures have different surroundings. In this method, we profit from this knowledge. If an edge $\{\mathbf{i},\mathbf{j}\}$ exists, then the pairwise term is defined as the similarity between descriptors of these vertices,

$$\psi(\mathbf{i},\mathbf{j}\ ;\ \mathcal{V}) = exp\left(\frac{-\|\mathbf{d}_i - \mathbf{d}_j\|^2}{\sigma^2}\right). \tag{6.3}$$

**Plane-like surface.** Here, the edges between different faces that can easily make a new face just by merging these faces together (they lie on one plane, or close to the plane) are preferred to have the same label. To achieve this, we benefit from the triangular meshes. Vertices and edges make triangular faces on the shape, most of the edges are members of two faces (except borders). If the edge connects two faces, the unary term is defined as cosinus of the angle between these faces (fig. 6.3. It can be easily computed as the normalized dot product of the face normals,

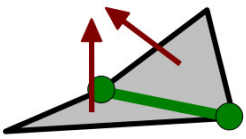


Figure 6.3: **Intuition behind plane-like surface pairwise term.** An edge (green) that connects two vertices supports to set the same label to vertices it connects (green spheres) if normals (red arrows) of two corresponding faces (gray triangles) are parallel.

$$\psi(\mathbf{i},\mathbf{j}\ ;\ \mathcal{V}) = \frac{\mathbf{n}_{ij}^{(1)} \cdot \mathbf{n}_{ij}^{(2)}}{\|\mathbf{n}_{ij}^{(1)}\|\ \|\mathbf{n}_{ij}^{(2)}\|}, \tag{6.4}$$

where $\mathbf{n}_{ij}^{(1)}$ is the normal of the face that has edge $\{\mathbf{i},\mathbf{j}\}$, $\mathbf{n}_{ij}^{(2)}$ is the second one.

## 6.4 Experiments

In the experimental section, we introduce alternative approaches to compute the unary term in §6.4.1. Then, parameters, datasets and descriptors are discussed in §6.4.2 and finally, we presents the labeling results in §6.4.3.

### 6.4.1 Unary term competitors

In addition to the proposed computation of the unary term, we evaluate results using the following methods based on the popular frameworks for 3D/2D recognition:

**VW:** In the training part, we assign all descriptors of the training shapes to the closest visual word from the vocabulary and we keep the histogram of occurrences of visual words in the class.

The normalized histogram of occurrences of visual words let us know the probability if the visual word is from class or not, we use this value as the definition of the unary term.

So, we have a scene's vertex, then we compute its descriptor, after that, we find the closest visual word and the pairwise term is defined by the corresponding value for the visual word in the normalized.

**VW-tfidf:** Here, we modified the method in *VW* by using term frequency-inverse document frequency (tfidf). In the original tfidf [171], visual words which are unique for the document (image) are preferred. We weight high visual words from the class $l$ instead other classes, so that, all class shapes were used as the first document and other shapes as the second document. The result is that we favor class $l$ is unique visual words instead of visual words from other classes. Tfidf is described in the paper of Sivic and Zisserman [171] in more detail and in the §2.1.4 of this thesis.

**Multi-VW:** Again, it is similar to *VW*. The difference is that during the training stage we perform multiple assignments instead of a single one. The training procedure is as follows, every vertex from the training set is represented as the set of all visual words closer than the threshold, instead of only the closest one. Using this, we will again obtain a histogram representing how often are specific visual words in the class used to construct the unary term. Similar to Lehmann *et al.* [102], multiple assignment is used only for the training part.

**Desc-NN:** The method is motivated by Boiman *et al.* [18] using the nearest neighbor in the descriptor space for class recognition. This method differs as it does not rely on quantization of visual words and does not include supervised learning of classes. We collect all descriptors for a specific class Then, the unary term for the vertex **i** is defined as the weighted distance to the closest

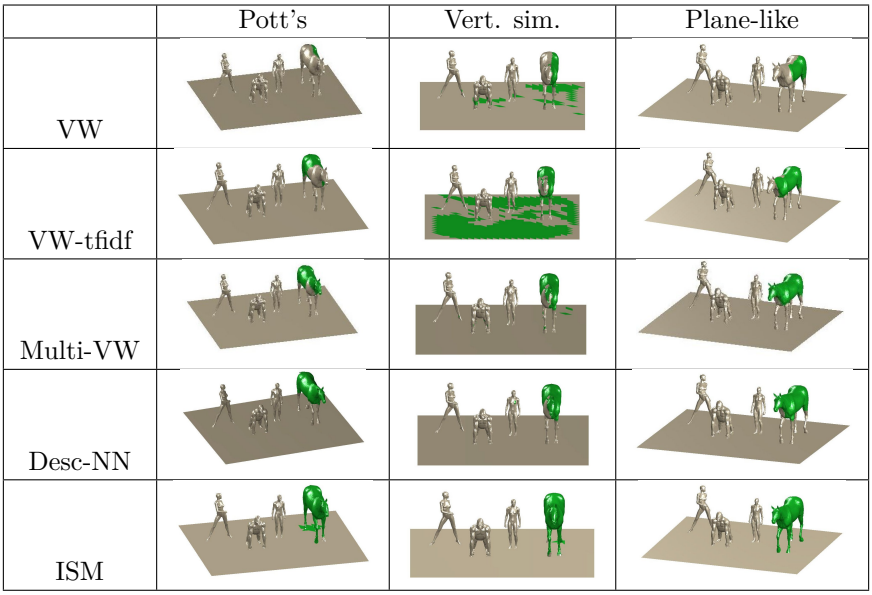



Figure 6.4: **Comparison of different methods to define unary and pairwise term.** Horse-class detection experiment run with weight $\phi = 1.3$ which was found as the compromise for all methods for this scene. Note that only ISM is able to better segment the object.

descriptors from the set of all class-specific descriptors,

$$\xi(\mathbf{i}\ ;\ \mathcal{E}, \mathcal{T}^c) = \exp\left(\frac{-||\mathbf{d}_i - \mathbf{d}_c^*||}{\sigma^2}\right), \tag{6.5}$$

where $\mathbf{d}_c^*$ is the closest descriptor for $\mathbf{d}_i$ from the set of all descriptors of the class. Here, $\sigma$ was set to 0.17, descriptors vectors have $L_2$ norm equal to one by default.

### 6.4.2 Parameters setup

**Terms normalization** Both unary and pairwise terms were normalized. As is common in graph cut algorithms [19], after the normalization, unary term was weighted by the $\alpha$ parameter and pairwise term by the $\beta$ parameter. Then, the weighting is defined as the unary term against the pairwise term, $\phi = \alpha/\beta$.

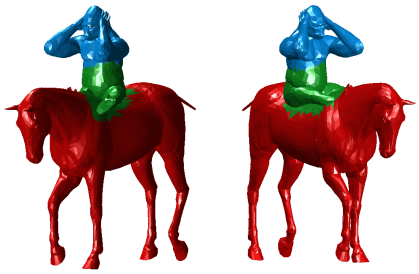


Figure 6.5: **Collision of the object detection.** The example shows two different classes connected to one shape. **Blue** color indicates the gorilla-class faces, **red** represents the horse-class and **green** shows faces which were labeled as the horse-class as well as the gorilla-class.

**Shape Descriptor** In our experiments we used modified 3D SURF [188, 85] computed at each vertex with a constant scale. Here, 3D SURF's feature is defined on the shape's vertex and the scale was set as a constant. Please note that the descriptor is not scale invariant anymore, but it allows a more dense shape description. We also experimented with a HKS [177] descriptor, but we found the best scene segmentation results for 3D SURF, however the experiment is not shown in this chapter.

**Datasets** We used the popular Tosca dataset [20] to define the training data. Then query scenes were made manually by giving several different objects into one shape. All objects in the query scene were also connected together and so the final scene was one connected mesh.

Additionally, we downloaded the scene from the 3D lighting contest [5] which represents a real life example. One of the shapes from Tosca that was not used for training was added to this scene.

### 6.4.3 Results & discussion on synthetic data

We applied the above described method to segment class specific objects in scenes using the set of shapes of the same class.

Fig. 6.4 compares different methods for the pairwise and the unary term. Note that ISM takes significant advantage from the training data as it models how the shape of the specific class should looks like and compare this model with the current shape. Competing methods suffer because their unary term is based

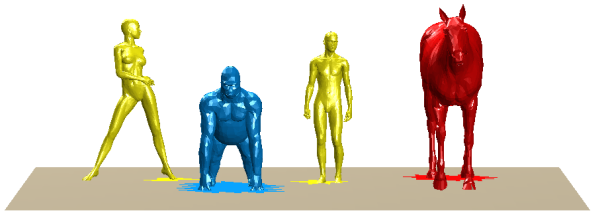


Figure 6.6: **Scene segmentation.** Here, we used ISM to compute unary term and Pott's method defined the pairwise term. Weights were set to $\phi = 1.3$ in this case. **Yellow** color represents the person-class, **blue** represents the gorilla-class and **red** represents the horse-class.

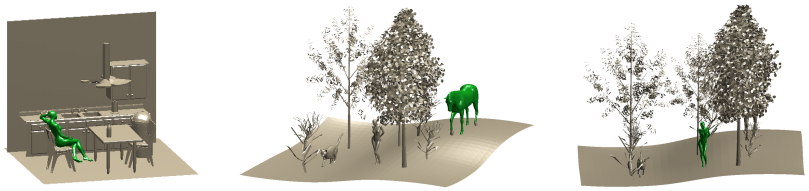
only on the statistical appearance of the vertex and pairwise term cannot handle full class structure.

Fig. 6.5 shows the collision of segmentation when two classes are close to each other. We can easily see that there is a colliding part recognized as both classes gorilla and horse. One can argue that gorilla's leg should be recognized as only gorilla, but using this result we know that gorilla's leg is close to the horse, and even more: we know which part of the horse was recognized as a gorilla, so the algorithm estimated where the gorilla is on the horse. This is an important result for further research.

Object detection/segmentation results are shown in fig. 6.6 and fig. 6.7 on the experimental scenes. Please note that we are able to recognize the class with a simple descriptor. The evaluations carried out demonstrate that the proposed method can successfully do joint class-specific detection and segmentation in cluttered scenes. The results we obtained are promising and indicate that further research along these lines will be fruitful.

### 6.4.4 Application on SfM data

As a final note, scenarios with reconstructing real-life scenes (PhotoTourism [173], Arc3D [186] ) and object localization/segmentation in that scenes [61] are very popular tasks these days. Even more, it has a huge impact for applications such as driving assistance or robot navigation systems. In this small experiment, we meet this problem of detection and localization in clutter and noisy real-life scenes.



(a) (b) (c)

Figure 6.7: **Scene segmentation results.** Several scenes merged with objects from the TOSCA dataset which were not used in training stage. (a) detection of the person class; (b) horse class; (c) and again person class. Though the rest of the scene was not use in the training it would still locally match to the object we want to segment. CRF+ISM yields to segment all local patches of trees etc. as a background.

Figure 6.8: **Examples of training shapes.** 6 out of 10 shapes used to train the class of bikes.

**Datasets** To investigate the performance on real-life scenes, a 3D mesh was created from a number of camera-shots around the object of interest (we experimented with a bike-class here) using the Arc3D [186]. 3D models were manually set to have the same scale and same bike orientation[1].

Ten scenes were selected for **training**. The background mesh was excluded and only bikes were kept for training. This creating process of training data is the only part when the user affect the process. Training data have 15K vertices in average (fig. 6.8).

For **testing**, we used the full reconstructions obtained from the Arc3D. The mesh was simplified to 130K vertices and directly used for our segmentation algorithm.

**Results of real-data segmentation** Qualitative results are shown in fig. 6.9. While shapes are extremely noisy, full of holes and incomplete, the combination of ISM (bike's location and how it looks like) and MinCut (segmentation algorithm) correctly segment the bike from background. The segmentation of bikes on the floor as well as extended experiments are for further research.

[1] Our detection/segmentation method is orientation and scale invariant by its definition, but the object has to be in a specific scale and orientation due to the feature descriptor.

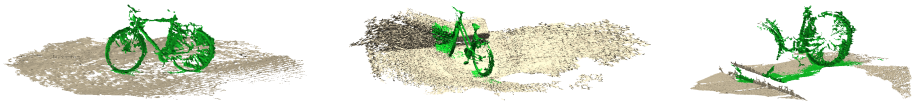


Figure 6.9: **Segmentation results.** Scenes were obtained using SfM algorithm from the set of camera-shots around the object of interest.

## 6.5 Conclusion

We have shown an approach for automatic class-specific objects detection in 3D scenes. The contribution is to combine segmentation with the object localization by fusing the top-down information provided by an object detection algorithm (ISM) with the bottom-up information of low-level scene priors (and data-dependent priors) which can be employed in GraphCut based segmentation. As opposed to traditional voting-propagation based segmentation in the original ISM work [104], this allows for the more efficient GraphCut to be used for segmentation. A vast amount of research in the 2D image community has employed joint recognition and segmentation of objects. This naturally inspires the use of similar techniques in 3D. Several possible methods for retrieval and classification using popular visual words or nearest neighborhood are modified for our task and compared with the presented approach. Object detection and segmentation results are shown on several scenes. We created test scenes by giving several different objects into one shape and segmenting object of the previously learn class. The parametrization of objects and mesh resolution are a contentious issue in determining the solution. While we assume the mesh is sensible enough to not affect our results in this work, this issue is worth more exploration.

# Chapter 7

# Joint object detection and segmentation

However, previous segmentation using an ISM based unary term introduced in Chapter 6 gives a number of promises; it has several weaknesses that makes it impossible to run in large-scale realistic scenarios:

(i) The algorithm requires a complete mesh where vertices are connected by edges.

(ii) The object detection algorithm is used to only initialize the segmentation algorithm and does not play a further role in segmentation which is done using CRF optimization.

(iii) The pair-wise term is defined by the scene and it is does not update during the iterations.

In this chapter, we overcome these weaknesses by efficient joint segmentation and detection of objects in large 3D scans of urban scenes. The method approximates message-passing in a model, which iterates between the layer of scene point labels and the layer of hypothesis for the object centers. The layers mediate correlations between points in a "shape-aware" way, allowing strong object shape priors to be enforced during inference. Message-passing is expressed as the voting and back-projection steps of a 3D ISM. We demonstrate that this approach improves upon the state-of-the-art for several object classes on a large urban point cloud.

The chapter is organized in the following way. The method is introduced in §7.1, and we discuss its relation to the state-of-the-art in §7.2. In §7.3 the method is described, in §7.4 we discuss parameters of the method, and we perform an evaluation in §7.5.

## 7.1 Introduction

Captured point clouds [30] are typically unstructured and unlabeled. They may be used for visualization, but preclude higher-level interactions requiring reasoning about *objects* in the scenes. For example a navigating robot may want to distinguish a fire hydrant from a car, or a home designer may want to virtually rearrange the chairs in a room. Hence, object recognition in 3D is becoming as important a problem as it is in 2D. In order to be generally applicable, one cannot assume that RGB images are also available: ground-level mobile mapping, for instance, is still dominated by LIDAR, which directly produces 3D point clouds.

To identify objects in point clouds, we must extract the shape of the object (segmentation) and assign it a class label, such as "car" or "traffic light" (recognition). Segmentation and recognition constitute a chicken-and-egg problem. The tasks are of mutual benefit: recognition is facilitated by knowing the shape of the object to be labeled, and segmentation is facilitated by knowing the object class. In 3D, however, the first-segment-then-recognize paradigm is still popular [61, 81, 185]. Approaches to recognize 3D objects without relying on an initial segmentation have only recently started to appear [83, 124, 178].

We now propose a novel approach for efficient joint segmentation and labeling of large 3D point clouds, using an approach that uses message passing. In our model, the interactions between points are mediated by an additional layer of variables representing possible object locations. Approximate inference in this model is performed via iterative message-passing. These messages are expressed as voting and back-projection steps. Each point casts votes for likely positions of the center of its parent object, and aggregating these weighted votes yields a distribution over object locations. Locations pass back messages, weighted by the confidence in the locations, to update label probabilities of points that voted for them, and the process repeats.

## 7.2 Background

For large 3D point clouds representing urban environments most work has focused on segmentation and recognition of large structures, such as buildings, roads, and trees. For recognizing small 3D objects in scans of cities, Golovinskiy *et al.* [61] proposed a pipeline where localization, segmentation, and recognition were performed in separate sequential stages, without feedback between stages. Velizhev *et al.* [185] took a similar approach, first segmenting the point cloud and then recognizing (possibly multiple) objects within each segment. These approaches fail when localization and segmentation heuristics perform poorly.

Additionally, joint segmentation and labeling of images is often posed as an inference problem in a probabilistic graphical model on image pixels, commonly a conditional random field (CRF) [99, 19, 168, 91, 76, 83]. The basic CRF formulation is natural for joint segmentation and labeling problems, in general. However, it does not recognize separate instances of an object class, and it does not take advantage of shape priors, which are an important aspects of our problem. For 3D point clouds of cities, distinctive local appearance features are rare, and yet the shapes of different instances within the same object class are highly consistent, and thus including priors on the shapes of objects is critical to good recognition performance. Work on higher-order potentials, hierarchical models, and densely-connected graphs has advanced the basic framework in this direction [65, 89, 97, 98, 91]. Campbell et al. [25] has proposed a shape prior with a simple Gaussian potential defined in a transformed descriptor space, but their approach is limited by dimensional constraints on the output space.

More complex shape priors encoding spatial relationships between parts have been proposed in part-based models, such as pictorial structures [48, 96]. For recognizing objects in 3D point clouds, a common approach is to first segment points based on geometric heuristics (e.g., remove the ground) and then learn a shape model from the segmented parts. For example, [81] learned a part-based model from training scans, deriving both part structures and deformation modes from unlabeled examples. [178] detected arrangements of scene elements after learning a part-based model from manually-curated training examples. [163] used a database of segmented models to reconstruct models from Kinect scans. The weakness of such methods is that they are highly sensitive to the quality of the initial unsupervised segmentation. Nan et al. [124] propose region-growing to construct segments obeying shape priors, but their approach is reliant on an initial over segmentation, extracts objects sequentially, and has not been shown to scale to large datasets.

Our method includes a random variable representing the probability of each

label being assigned to each input point, plus unary interactions between point labels and their observed properties. As in an ISM, new random variables are introduced in a hypotheses space representing the probability of an object with a particular label centered at that location. Pairwise interactions between point labels are mediated through the labeled object location hypotheses.

In contrast to standard ISM models [103, 83], ours maintains continuous probabilities for all object classes for all input points and all potential object centers. That is, all object labels are in play for all input points and all potential object centers at all times. Specifically, we do not perform non-maximum suppression to extract discrete sets of object centers, we do not perform discrete segmentation of input points based on vote back-projections, and we always ensure that the sum of probabilities for possible class labels (including the background) sum to one for all input points.

## 7.3 Method

We have a scene that consists of a set of 3D points. If $\mathbf{i}$ is a 3D point of the scene, it has position $\mathbf{p}_i$, feature descriptor $\mathbf{d}_i$ and assigned class label $x_i \in \mathcal{L}$ from a set of labels.

We also define the object hypotheses by Hough voting. Each vote cast $\boldsymbol{\lambda}$ is associated with:

(i) Position of the vote cast.

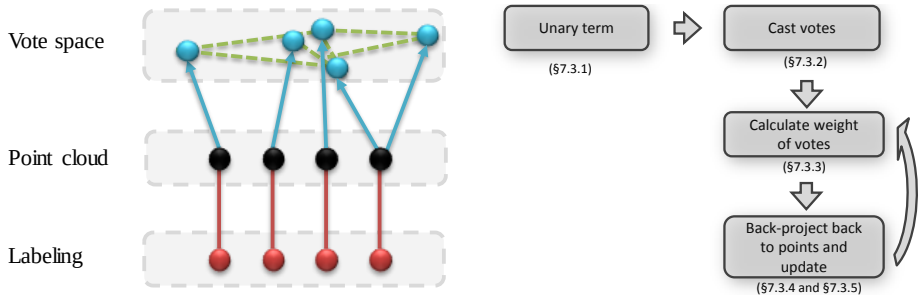
(ii) Point that casted the vote cast.

(ii) Probability that the 3D point that generated $\boldsymbol{\lambda}$ has the label $\ell$.

(iv) Density of vote casts at position $\boldsymbol{\lambda}$.

There are two types of edges in the model. The first type of edge connects space of 3D points with space of the vote casts (blue lines in fig. 7.1(a)). These edges are generated during the voting process and they are used to pass probabilities from points to votes and vice versa. These probabilities are described later in detail. For now, we just say that these probabilities correspond to how likely 3D points have a class label as well as how likely vote casts are a real center of any object.

Second type of edges densely connect the vote casts to other vote casts in their neighborhood (green dotted lines in fig. 7.1(a)). These lateral edges positively reinforce hypotheses that are judged to be equal (implemented as summing up




(a) Model (b) Flow chart of the method

Figure 7.1: **The method.** (a) Our model. It consists of a point cloud of 3D scene (black dots). The goal of the method is to estimate a class label of each 3D point (red dots). Points of the point cloud are not connected directly, but they are connected in the vote space via their vote casts (blue dots). Two 3D points likely belong to the same object, if their vote casts are close to each other (green lines). (b) Flow of the method. First, initial labeling is estimated. Second, we cast votes. Third, we update weights of the votes (based on how likely point cloud points belong to the class and their vote casts are close to each other). Fourth, the value of updated weights is back-projected to the point cloud and point's labeling is updated. Note that vote casts can be precalculated as voting depends only on the similarity in the descriptor space.

nearby votes via kernel density estimation), but negatively reinforce neighboring nodes, so that two distinct hypotheses for object center with the same label are not too close to each other, i.e. the corresponding objects do not overlap.

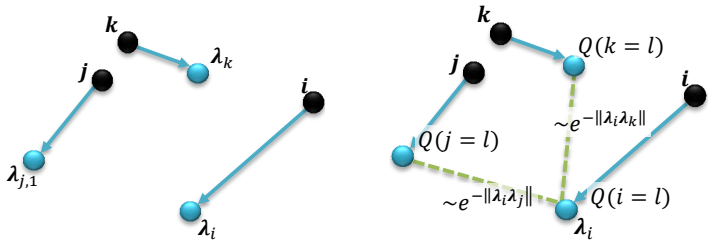
We perform approximate inference in the model described above to assign each point in a 3D scene a class label. We define a distribution $Q_i(\ell)$ approximating the probability for point $\mathbf{i}$ taking label $\ell$. Starting from an initial distribution computed by applying a classifier to each point independently, the algorithm iteratively updates $Q_i$ at all points by message-passing in the model.

The message-passing is illustrated in fig. 7.1(b) and can be broken into three steps.

(i) In the first message-passing step, points independently vote for the object hypotheses, based on their local descriptors.

(ii) In the second message-passing step, the strength of each hypothesis is reinforced by others at similar position by kernel density estimation.

(iii) Finally, in the third step, messages are back-projected from the hypotheses and the probabilities of points are updated.




(a) Voting (b) Evaluating the hypotheses

Figure 7.2: **Illustration of the method.** (a) Points of the point cloud (highlighted by black points) cast votes (highlighted by blue points). Each point has its approx. probability whether it belongs to the class. These probabilities are also associated with vote castes of the points. (b) For vote $\boldsymbol{\lambda}_i$, that was casted by point **i**, we update the hypothesis that $\boldsymbol{\lambda}_i$ is a center of the object. The update is estimated from the weighted distances to other centers (eq. 7.2). Once the hypothesis is updated, its value is back-projected to the 3D point that casted the vote.

Since we do not have a voting model for the background class, we estimate the background probability by normalizing the probabilities of other as described in §7.3.4.

### 7.3.1 Initialization

To initialize the algorithm, label probabilities, $Q_i(l)$, are assigned to each point independently. We learn a unary classifier that maps point descriptors to a probability for each label. We experimented with several classifiers such as nearest neighbor, SVM, Hough forest etc. In the end, JoinBoost [182] was found to work best.

### 7.3.2 Voting

We cast a set of votes $\boldsymbol{\Lambda}_i$ from the 3D position of each scene point $\mathbf{p}_i$. We note $\mathbf{v}_a^*$ a vector of the $a$-th training point to the center of the training object. Then, $i$-th test scene point cast votes from training data points with descriptors within distance $\delta$ of the descriptor $\mathbf{d}_i$:

$$\boldsymbol{\Lambda}_i = \{\mathbf{p}_i + \mathbf{v}_a^* \quad | \quad \|\mathbf{d}_i - \mathbf{d}_a^*\| < \delta \quad \text{and} \quad \gamma(i,a)\}\,, \tag{7.1}$$

where $\gamma(i,a)$ is a cutoff function that suppresses the vote from $a$-th training point with label $\ell_a^*$ if the current confidence that $p_i$ will be assigned this label

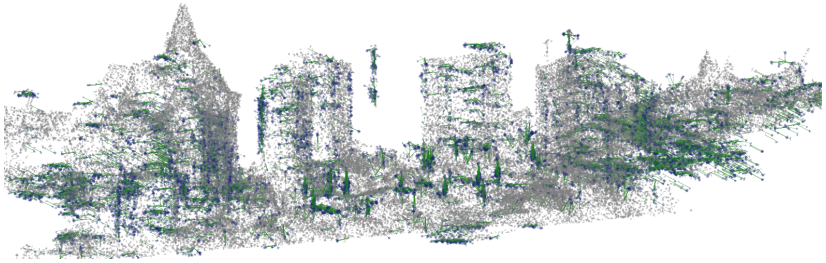


Figure 7.3: **Distribution of votes of the traffic light class.** Green lines show votes from the points and blue highlights the position of estimated votes. We expect a high density of blue at the centers of the objects.

is negligible. $\gamma$ slightly decreases false positives, speeds up the inference, and saves a lot of memory. Figure 7.2(a) illustrates the process of casting votes. Figure 7.3 shows the distribution of votes for the traffic light class on a part of our test scene.

It is obvious that most promising object centers are occupied by votes with high probability of being the object. Next section focuses how to calculate it.

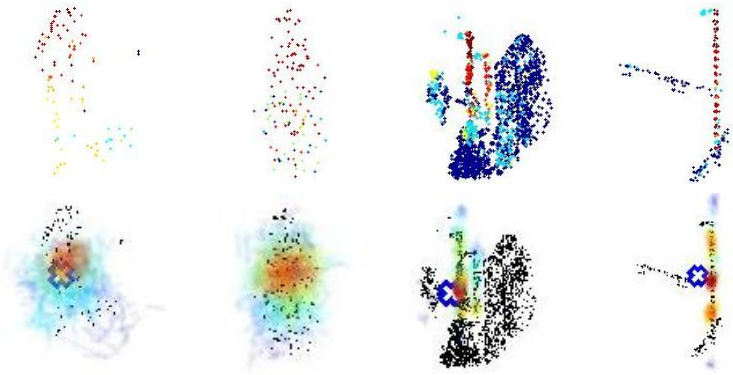
### 7.3.3 Evaluating Hypotheses

Every vote is likely to be a center of the object if there is a high density of votes for the same object center next to that vote. To be able to evaluate how likely is a vote cast a center of the object, we clarify the notation that connects 3D points with their votes. We note some $k$-th vote as $\boldsymbol{\lambda}_k$, we will refer to the point that casted $\boldsymbol{\lambda}_k$ as $x_{\boldsymbol{\lambda}_k}$. The $k$-th vote is now also associated with $Q(x_{\boldsymbol{\lambda}_k} = \ell)$ which refers to the approx. probability that point that casted $\boldsymbol{\lambda}_k$ belongs to the object-class $\ell$.

The probability that there is an object with label $\ell$ and center $\boldsymbol{\lambda}$ is computed by the density estimation:

$$p(\boldsymbol{\lambda} \mid \boldsymbol{\Lambda}, \ell) = \frac{1}{K \cdot w_\ell} \sum_{k=1}^{K} Q(x_{\boldsymbol{\lambda}_k} = \ell) \cdot \exp\left(-(\boldsymbol{\lambda} - \boldsymbol{\lambda}_k)^T \Sigma (\boldsymbol{\lambda} - \boldsymbol{\lambda}_k)\right), \quad (7.2)$$

where $\boldsymbol{\lambda}_k$ is the $k^{\text{th}}$-closest vote to vote $\boldsymbol{\lambda}$, $Q(x_{\boldsymbol{\lambda}_k} = \ell)$ is the current approx. probability that the voting point $x_{\boldsymbol{\lambda}_k}$ has label $\ell$, $\Sigma$ is the covariance matrix for density estimation, and $w_\ell$ is a normalization factor to account for noise in the



(a) AK-TP (b) AK-FP (c) SL-TP (d) SL-FP

Figure 7.4: **Vote space.** The first row shows the probability of points having a class label, $Q()$, it goes from blue for low to red for high. The second row shows votes and their color represents the value that will be back-projected (combination of vote's density and $Q()$ of points cast). The truth center is highlighted by a blue cross. We show true positive (TP) results and false detection (FP) for two classes: AdKiosk (AK) and StreetLight (SL). Note that wrong hits (b,d) are caused by the similar geometry between different classes. The proposed method infers across all possible labeling and point combinations and finds the optimal solution.

voting process. The $\Sigma$ and $w_\ell$ are learned by cross-validation as described in §7.4. The density is estimated over $K = 200$ votes. Figure 7.2(b) illustrates the process of hypothesis evaluation. Figure 7.4 shows votes for four objects and the approx. probability that will be back-projected to the 3D points.

### 7.3.4 Back-Projection

After establishing the distribution over object hypotheses in the vote space, we update the label probabilities at each 3D point. The update equation assigns each 3D point the probability of the best vote it casted:

$$Q_i(x_i = \ell) \;=\; -\exp\Big(-\max_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}_i} p(\boldsymbol{\lambda} \mid \boldsymbol{\Lambda}, \ell)\Big) \tag{7.3}$$

After updating the probabilities, we normalize them. When doing this, we must also take into account the background class, for which we have no object model beyond a unary classifier. Our procedure for doing this is described below.

### 7.3.5 Handling the Background

For normalizing the probabilities for different labels at each point, we must also consider the probability that the point belongs to the background. Thus, we first estimate, for each label $\ell$, the probability that the point does *not* have label $\ell$:

$$Q_i(x_i \neq \ell) \;=\; -\exp\left(-\psi_1^{\text{bcgr}}(x_i) - \max_{\ell' \in \mathcal{L}\setminus\ell} \tilde{Q}_i(x_i = \ell')\right) \tag{7.4}$$

Now, we normalize the probability for each label as

$$Q_i(x_i = \ell) \;\longleftarrow\; \frac{Q_i(x_i = \ell)}{Q_i(x_i = \ell) + Q_i(x_i \neq \ell)} \tag{7.5}$$

The revised probability that the point belongs to the background is then $1 - \sum_{\ell \in \mathcal{L}} Q_i(x_i = \ell)$.

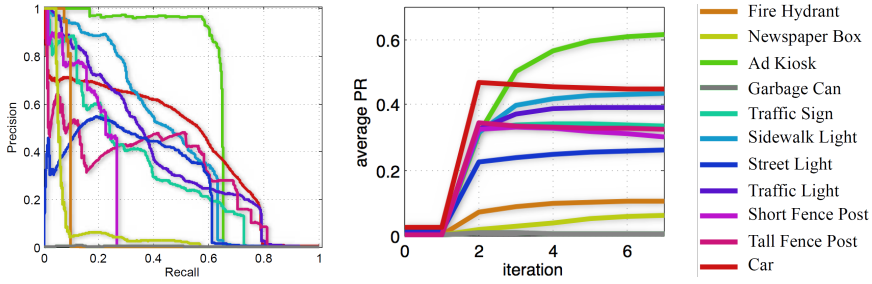
## 7.4 Implementation Details

In this part of the thesis, we did not use 3D SURF features because we wanted same descriptors as our competitor [61]. As the first step of learning the object model, descriptors are defined for each point in the scene. These descriptors consist of spin images [74] computed with four shells and five slices (20 values total) in a cylinder of radius 2m and height 5m centered at the point $p_i$ and oriented along the global up vector.

Then, a JointBoost classifier [182] is trained on the descriptors and labels of the training data. The output of this classifier is the unary term, i.e. the initial label probabilities predicted by the model without taking correlations between points into account.

Next, a data structure is constructed for Hough voting. Given the descriptors of a point, the structure returns the set of Hough votes, for each label, that should be cast from it. We experimented with Hough forests [55] but found that a $k$-nearest neighbors query structure [8] yielded better results for our dataset.

Hough voting algorithms have several parameters which are usually hard to set up and they affect performance significantly [103, 83, 102]. We estimate these parameters automatically by cross-validation. The training data is split into two parts: validation training and validation test. The system is trained on the first part and parameters estimated on the second. As some parameters affect others, the estimation of these parameters was run as a cascade. There




(a) (b) (c)

Figure 7.5: **Performance of scene recognition.**(a) Per-class precision and recall for scene segmentation. (b) The advantage of iterations. Average area under the precision-recall curve increases with iterations. (c) Legend.

are three parameters that were pre-calculated: $\delta$, $\Sigma$, and $w_l$. Let us describe their estimation in detail.

Parameter $\delta$ represents a minimal distance between point descriptors (eq. 7.1). It is estimated as the maximum distance that is needed to cast votes from validation test data using the validation training data.

Parameter $\Sigma$ represents a covariance matrix for kernel density estimation in vote space (eq. 7.2). It is calculated from the variance of vote casts for the centers of single objects in the validation test set. The covariance usually reflects the principal directions of variation of the object class: we observed that this parameter automatically learn that there is a high variation in the horizontal direction for streetlights, while the major variation for the car class is in the $xy$-plane.

Parameter $w_l$ is a normalization factor for aggregated per-class vote strengths at a point (eq. 7.2). Since we do not cast votes for the background class, we cannot simply normalize the foreground vote strengths by dividing by their sum. Thus, $w_l$ is learned by cross-validation. We assume that recognizing testing data cannot be better than recognizing training data. For every training object, we build a voting structure on the other training objects, and cast votes for the center of the selected object. Then, $w_l$ is estimated as $\sup Q_i(x_i)$ over all training objects.

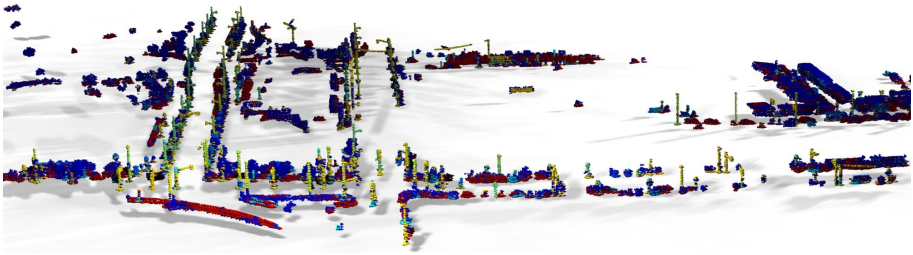


Figure 7.6: **Visualization of segmentation and labeling results on Ottawa dataset.** Each color corresponds to a class. Points detected as background are not shown.

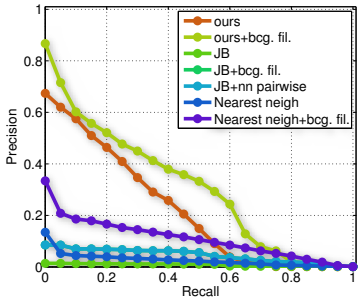
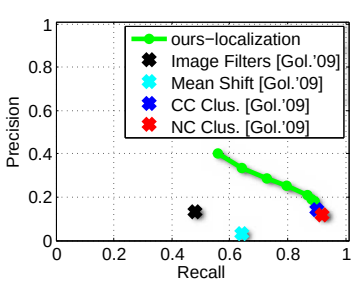


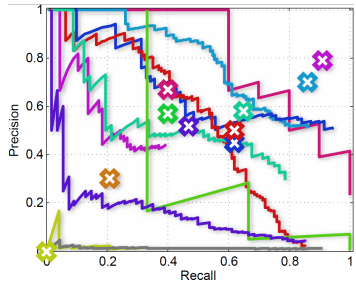

Figure 7.7: **Average precision/recall for scene segmentation over all classes.** We compared with JointBoost [182], nearest neighbor [18] and graph-cuts [19], each with and without pre-filtering out background regions [61, 185]. Our approach outperforms competing methods even without background filtering.

## 7.5 Experiments

We tested our segmentation and labeling algorithm on a LIDAR scan covering 0.3km$^2$ of the city of Ottawa, Canada [125, 61, 185]. The original dataset has about 100 million points — for our experiments we down sampled it to 5 million points, although descriptors were computed on the full-resolution data. The scene has ground truth in the form of segmented and labeled man-made objects, ranging in size from fire hydrants to vans. The truth objects constitute 11 foreground classes (fig. 7.5).

The results of our segmentation and labeling algorithm, evaluated on the test area of the dataset, are presented in figure 7.5 as class-specific precision-recall curves. In figure 7.7, we present the average precision-recall over all classes, compared to results obtained using (a) only a unary classifier — JointBoost [182] and feature-based nearest neighbors [18] — at each point, and (b) using graph cuts in a simple CRF model with the pairwise term defined as a Gaussian on the distance between point descriptors. Our method clearly outperforms these simpler models. Further, we present the improvement in results if we




| Class | Pr [61] | Rec [61] | Pr (our) | Rec (our) |
|---|---|---|---|---|
| Tall fence post | 0.67 | 0.40 | 1.00 | 0.40 |
| Sidewalk light | 0.70 | 0.86 | 0.49 | 0.86 |
| Street light | 0.45 | 0.62 | 0.53 | 0.61 |
| Short fence post | 0.79 | 0.91 | 0.47 | 0.42 |
| Car | 0.50 | 0.62 | 0.40 | 0.62 |
| Traffic sign | 0.58 | 0.65 | 0.44 | 0.64 |
| Ad kiosk | 0.00 | 0.00 | 1.00 | 0.33 |
| Traffic light | 0.52 | 0.47 | 0.13 | 0.47 |
| Newspaper box | 0.00 | 0.00 | 0.00 | 0.00 |
| Garbage can | 0.57 | 0.40 | 0.01 | 0.40 |
| Fire hydrant | 0.30 | 0.21 | 0.00 | 0.00 |

Figure 7.8: **Comparison of object localization and detection to Golovinskiy *et al.* [61].** As our method considers every point in $\mathbb{R}^3$ a possible object center (with varying probability), we perform non-maximum suppression and sort the maximum by their probability, to compute precision-recall curves. Top-left: Localization results of our (green curve) are competitive with the best methods proposed by Golovinskiy et al. Top-right: Detection results (localization per class). The crosses represent the PR values reported by Golovinskiy et al. Bottom: Optimal PR values for detection. Since our method tries to keep all possible hypotheses in play during iteration, it is tweaked for high recall and hence does slightly worse in precision on several classes.

apply a heuristic to pre-filter out background regions (large planar areas) as in Golovinskiy et al. [61] and Velizhev et al. [185]. While our method is designed to detect background regions by default, it benefits from this prior filtering since the background (buildings, roads, vegetation) is extremely heterogeneous and accounts for most of the scene area.

Figure 7.5 demonstrates the benefit of the iterative optimization procedure. The area under the precision-recall curve increases for all classes as we iterate message-passing.

Finally, although this is not a focus of our work, we present a comparison of localization (identifying object centers irrespective of class) and detection (identifying object centers and assigning them the correct class label) with the results of Golovinskiy et al. [61] on the same dataset (fig. 7.8). For parity with

their approach, we also filtered out probable background regions in advance using the identical method. Our method does not explicitly compute discrete object centers, instead maintaining a probability distribution over all possible centers. We performed non-maximum suppression to identify maximum in the distribution as candidate centers, and sorted them by their probabilities to obtain precision-recall curves for localization and detection. Our method is competitive to that of Golovinskiy et al. which is specifically designed for this task, and improves upon it for some classes, even though this is only an intermediate result of our optimization procedure.

## 7.6 Conclusion

In this chapter, we have introduced an efficient method for joint segmentation and detection of objects in a large 3D scans of urban scenes. The method approximates message-passing in a model which iterates between the layer of scene point labels and the layer of hypothesis for the object centers. Message-passing is expressed as the voting and back-projection steps of a 3D ISM. We demonstrate results on the LIDAR scan of Ottawa city.

# Chapter 8

# Shape Completion

In Chapter 7, we have already shown how to find objects in noisy scenes as well as how to segment objects from the background in city-scale scenes. These scenes are usually noisy and contain a lot of missing data. In this chapter, we investigate the possibility of combining the detection results with the information that is stored in 3D datasets to complete parts of the noisy 3D scene.

Previously, shape completion approaches were attempted in carefully tailored datasets, with user guidance and, strong assumptions, making generalization difficult. When the problem was tackled by object-specific 3D ISM (we will discuss its relevance to the nearest neighbor method), it suffers from generalization weakness as it is only object specific. Thus, motivated by recent advances in deep learning that gives great results at large-scale, data-driven, unsupervised learning, we show how methods using Restricted Boltzmann Machines (RBMs) can automatically glean structure to generate and complete shapes effectively. We substantiate observations and insights with comprehensive, reproducible experiments on the Huang shape dataset [70]. Though our method completes voxelized shapes, the robustness of this method to deformations and large holes, makes future integration with recognition and reconstruction frameworks a viable possibility, paving the way for more semantic vision. Then, we show how to use this completion method to glean a structure of certain parts of the noisy 3D scene.

We introduced the problem and discussed related work in §8.1. Then, we introduce RBM and its application to completion in §8.2. We then trained and evaluated a variety of shape completion competitors to study performance and summarize our observations in §8.3.2. Finally, we demonstrate in §8.4 how to apply the proposed completion technique to improve objects of noisy large-scale

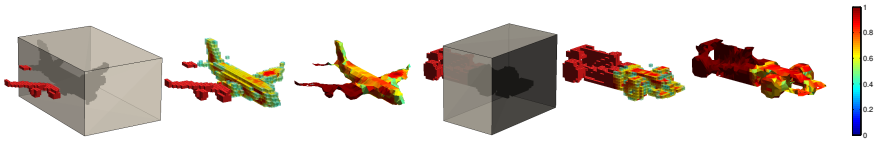


Figure 8.1: **Shape completion.** Object structure is automatically learnt to complete extremely occluded objects in the two examples above. The missing regions are highlighted by cubes in columns 1,4 and are completed in voxel representation in columns 2,5 (meshed results are also plotted in columns 3,6). The hypothesized voxel's color represents its probability: ranging from blue for 0 to red for 1.

scenes where results of detection/segmentation Chapter 7 are used to initialize the completion.

## 8.1 Introduction and related work

We hope to achieve integrated shape reconstruction and completion ultimately, but addressing purely 3D data first, creates a generic solution applicable to any shape (and across representations).

Theoretically, shape completion has been a captivating problem, even when 3D data was less abundant. There are numerous difficult questions. What is the scale and location of the hole? How do we define a golden parametrization and match the faulty test shape against it? How can one be recognizant of deformation but account for viewpoint invariance in this process? Training data often lacks any perfect, golden examples and the tediousness of 3D annotation defines the need for generic, automated, robust methods with minimum manual tuning or annotation.

Methods for 3D shape completion [136, 75, 51] are based on mesh fitting, while it is still hard to learn dependencies between parts [75] for meshes and there is nothing like a community established completion benchmark. We counter this by firstly representing our meshes in a more discretized framework (as Kinectfusion [72], 3D CRFs [78]) and additionally by employing the state of the art techniques from learning that have shown good modeling power [47]. We also introduced a novel shape completion benchmark.

A slew of methods [39, 162, 6, 136, 71, 137, 51] attempted completion by various approaches. The choice of shape representation significantly impacts the method and results. Having a strong golden parametrization [6] or simple manifolds and assistance with markers or boundary constraints [158], help in

reliable matching and correspondence finding. Mesh transplantation [175, 71] is challenging and needs identifying the transplant and the translate, rigid mapping of shapes, topologically corresponding (homeomorphic) boundaries for stitching, a common domain parametrization for the surface regions. Finally stitching can be performed with some user assistance ensuring that there is no "bursting at the seams". These systems are rarely available to try or easily reproduce, making it hard for creating solutions that will work out of the box for freeform 3D shapes.

In contrast, image completion is well explored. The 2D pixel driven parametrization of images has allowed a variety of reproducible completion and texture synthesis algorithms [152, 90] some even simultaneously estimating other transformation parameters [41]. The typical machine learning algorithm relies on feature representation (with corresponding dimensions assumed to be in implicit correspondence) and this fits nicely with discretized image representation. Naturally, completion and inference schemes can be easily extended to Monge-patches [158] or voxelized schemes [78]. Recently, the voxelized shape representations significantly improve image recognition [78] as well as they make a good choice for learning RBM [47].

**Semantic scale:** Shape completion approaches can be distinguished by the scale at which they model shape. Very local methods for shape completion, including [39, 24] are sophisticated interpolation techniques, effective for small holes, but not for learning meaningful semantics. Methods including [137, 162] advance this by using parts of the same shape (based on self-similarity) for completion. In contrast, global methods sometimes need a global parametrization [6] or model the shape variation from large, clean datasets [136], neither of which can be guaranteed. In this respect, the RBM framework is powerful, the higher order potential terms automatically figure out the scale of shape semantics lie, automatically and effectively [47].

**Graphical models:** Typically scene understanding problems in vision, rely on a supervision phase, trying to convert every intuition and heuristic into sophisticated feature representations and complex learning schemes. With increasing data (making dense annotation difficult) and improved computational resources, the area of semi-supervised learning is increasingly important. Improved computational resources and approximate optimization techniques in neural network algorithms led to deep learning algorithms [67, 92, 100] that were not only working but beating established benchmarks set by their supervised counterparts in various vision tasks.

Graphical models are being increasingly used to solve problems in 3D ranging from local shape models in Monge-patch representations [158] to learning context in sparse 3D representation [11]. As nicely motivated by [47], RBMs model specific families of higher order potentials on fully connected graphs, which allow for simple and effective inference. Thus, the scale of the structure learnt, is left to the algorithm in a very data-driven way.

## 8.2 The method

**Problem Statement:** Given a dataset of voxelized 3D models, and a partially observed voxelized test model we want to predict the incorrect/missing regions. In this process, we assume that the region of the hole is known and thus we know which part of the shape is given and which part is unknown. The RBMs are generative models that learn the hidden features (and relationships) in data in an unsupervised manner. These can be then used for generative purposes or to conditionally predict the unobserved nodes given the observed ones.

### 8.2.1 Energy-based models

Energy models associate a scalar energy to each configuration of the variables. The model has a form of energy function: $E(\mathbf{v};\theta)$, where $\mathbf{v}$ is a vector of states of the model and $\theta$ represents its parameters. Learning the energy-based models correspond to modifying that energy function. The goal of learning is to find parameters that minimize the energy, this corresponds to the maximization of the probability,

$$p(\mathbf{v}|\theta) = \frac{\exp\big(-E(\mathbf{v};\theta)\big)}{Z(\theta)}. \tag{8.1}$$

This equation includes the normalization factor, $Z(\theta) = \int_{\mathbf{v}} \exp(-E(\mathbf{v};\theta))d\mathbf{v}$. The normalization factor is called a partion function. The partion function is hard to calculate for real applications as it needs to estimate $E(\mathbf{v};\theta)$ for every possible state of the model. In the following text, we will review Restricted Boltzmann Machines and show how to avoid calculating the partion function.

### 8.2.2 The basic RBM

An RBM [172] is a bi-partite graph with a visible layer $\{\mathbf{v}_j|j \in \{1 \dots J\}\}$ and a hidden layer $\{\mathbf{h}_i|i \in \{1 \dots I\}\}$, such that each node in $\mathbf{h}$ is connected to every node in $\mathbf{v}$ and vice versa (no connections within a layer). Graphical

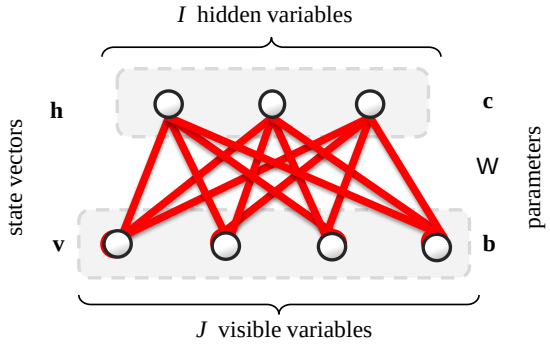



Figure 8.2: **RBM model.** Restricted Boltzmann Machine is an undirective graph of nodes separated into two distinct layers. The top layer of nodes that represents hidden random variables, **h**, and the bottom layer of nodes that represents visible variables, **v**. While the visible nodes are for observed data, the second layer of hidden nodes reveals some unknown representation of the data. Every node is restricted to have connections to nodes in the different layer only. This yields to further effective computations, because the state of the hidden variables can be estimated from the state of the visible variables only, and vice versa. There are three latent variables that has to be estimated during the learning. The bias of each visible node, **b**, the bias of each hidden node, **c**, and the weight of every connection, W. During the learning phase, we want to estimate latent variables such that the RBM generalizes on the training set of data. Note that nodes in the Boltzmann Machines (BM), in contrast to this restricted version, connect to nodes within the same layer too. Thus it is harder to learn BM than RBM [66].

representation of the RBM is shown in fig. 8.2. The joint energy and probability are defined as:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{j=1}^{J} b_j v_j - \sum_{i=1}^{I} c_i h_i - \sum_{ij} h_i w_{ij} v_j \quad \text{where}\, \theta = \{\mathbf{b}, \mathbf{c}, \mathsf{W}\}, \tag{8.2}$$

$$p(\mathbf{v}, \mathbf{h}|\theta) = \frac{1}{Z(\theta)} \exp\big(-E(\mathbf{v}, \mathbf{h}; \theta)\big) \tag{8.3}$$

Given a set of examples at the visible layer, the RBM can learn parameters to model hidden features/relationships for this data. The *visible* layer represents shape data voxels, regardless of whether they are surface, background or hole. The *hidden* layer represents specific configurations of the visible nodes, but what they really learn is usually unknown (it is why they are called hidden). Latent parameters of the model, $\theta$, learn during the training.

To understand RBM training, we introduce the joint, marginal and conditional distributions. We show how RBM sampling is performed. Given these tools, we finally give the algorithm for training and sampling from the RBM.

**Marginal:** The hidden nodes are never observed, so the marginal distribution of the visible nodes is obtained by integrating the joint probability:

$$p(\mathbf{v}) = \frac{1}{Z} \exp\left( - \left( -\mathbf{b}^\top \mathbf{v} - \sum_i \ln \left( \int_{h_i} \exp\left( h_i . \left( c_i + \mathbf{W}_i . \mathbf{v} \right) \right) dh_i \right) \right) \right). \quad (8.4)$$

Inspired from physics, the notation of free energy, $F(\mathbf{v}; \theta)$, is introduced. This free energy maps formulation with visible and hidden variables to the formulation with visible variables only, thus the formulation is similar to one in §8.2.1,

$$p(\mathbf{v}|\theta) = \frac{\exp\big(-F(\mathbf{v};\theta)\big)}{\int_v \exp\big(-F(\mathbf{v};\theta)\big) d\mathbf{v}}. \quad (8.5)$$

Free energy is then defined as,

$$F(\mathbf{v};\theta) = -\mathbf{b}^\top \mathbf{v} - \sum_i \ln \left( \int_{h_i} \exp\left( h_i \cdot \left( c_i + \mathbf{W}_i . \mathbf{v} \right) \right) dh_i \right). \quad (8.6)$$

**Conditional:** Because of the RBM structure the conditional distribution obeys: $p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h}) = \prod_j p(v_j|\mathbf{h})$. Further the conditional distribution takes the form:

$$p(\mathbf{h}|\mathbf{v}) = \frac{p(\mathbf{h}, \mathbf{v})}{p(\mathbf{v})} \quad (8.7)$$

$$\Rightarrow p(h_i|\mathbf{v}) = \frac{\exp\left( h_i^\top \cdot \left( c_i + \mathbf{W}_i \mathbf{v} \right) \right)}{\int_{h_i} \exp\left( h_i^\top . \left( c_i + \mathbf{W}_i \mathbf{v} \right) \right) dh_i}. \quad (8.8)$$

**Sampling:** Sampling from an RBM is useful for two reasons. First, it gives an idea of what the model has captured. Second, it is useful in training, because the accurate estimation of the distribution $p(\mathbf{h}, \mathbf{v}|\theta)$ still have a problem that hidden nodes have to be estimated from visible nodes and vice versa. Thus one can iteratively estimate $\mathbf{h}$ given $\mathbf{v}$, and then a new $\mathbf{v}$ given $\mathbf{h}$. This process is called sampling. Given a joint distribution of N random variables $\mathcal{S}$, Gibbs sampling performs sampling sub-steps of the form: $S_i \sim p(S_i|S_{\sim i})$. Because of the bi-partite nature of the RBM, a hidden node is independent of the others

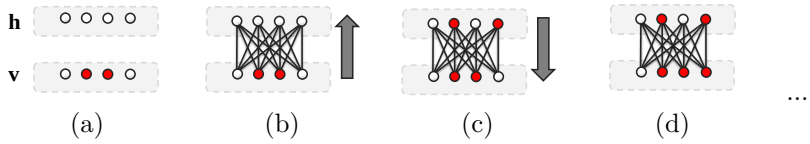



Figure 8.3: **Illustration of sampling.** Visible and hidden nodes are sampled. Given a known state of visible variables (a), hidden variables can be estimated (b). From these hidden variables, we can re-estimate visible variables (c,d). For sufficiently large values of samples, final sample will be uncorrelated with the original image [47]. But we have observed that three to four samples are enough [47, 92].

given the visible layer (and vice versa). Conditional samples in the Markov chain are gathered sequentially as:

$$\mathbf{h}^k \sim p(\mathbf{h}|\mathbf{v}^k), \text{ and } \mathbf{v}^{k+1} \sim p(\mathbf{v}|\mathbf{h}^k). \tag{8.9}$$

From a starting configuration, sampling is performed and as $k \to \infty, (h^\infty, v^\infty)$ would form accurate samples of the distribution $p(\mathbf{h}, \mathbf{v}|\theta)$. The process of sampling is illustrated in fig. 8.3.

### 8.2.3 Training the RBM

We are given data in the form of visible node layers $\mathcal{D} = \{\mathbf{v}_n | n \in \{1 \dots N\}\}$, and our goal is to learn the graph parameters $\theta$ in a maximum likelihood framework [66] (while marginalizing the the hidden nodes) so that:

$$\theta = \underset{\theta}{\operatorname{argmax}} \prod_n p(\mathbf{v}_n) \equiv \underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} \left(-\ln p(\mathbf{v}_n)\right) \tag{8.10}$$

Due to the lack of a closed-form solution, the ML estimate of $\theta$ is learnt using a specific type of the stochastic gradient descent called contrastive divergence. For $p(\mathbf{v}) = \exp(-F(\mathbf{v}))/Z$, the gradient for the negative log likelihood is:

$$-\frac{\partial \ln\left(p(\mathbf{v})\right)}{\partial \theta} = \underbrace{\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{v}, \mathbf{h}\right)}{\partial \theta} | \mathbf{v}\right]}_{\text{Positive phase}} - \underbrace{\mathrm{E}_{\mathbf{v},\mathbf{h}}\left[\frac{\partial E\left(\mathbf{v}, \mathbf{h}\right)}{\partial \theta}\right]}_{\text{Negative phase}} \tag{8.11}$$

This gives us direction on how to update RBM parameters. However it seems obvious that the naming of these two terms, positive and negative, refer to their signs, the naming is taken from their effect on the probability density. Positive term increases the probability of training data we want to learn by reducing the

free energy. The negative term reduces probability when the RBM generates data by itself.

To estimate these term of the positive phase and term of the negative phase, we can substitute $\partial E\left(\mathbf{v},\mathbf{h}\right)/\partial[b_j,c_i,w_{ij}]$ by $[-v_j,-h_i,-h_i.v_j]$ in (8.11). Then the positive phase is:

$$\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{v},\mathbf{h}\right)}{\partial\{\mathbf{b},\mathbf{c},\mathsf{W}\}^{\top}}|\mathbf{v}\right]=\left[-\mathbf{v}^{\top},-\mathrm{E}_{\mathbf{h}}\left[\mathbf{h}|\mathbf{v}\right]^{\top},-E_{\mathbf{h}}\left[\mathbf{h}\otimes\mathbf{v}^{\top}|\mathbf{v}\right]\right] \tag{8.12}$$

While the positive phase is usually tractable, the double expectation of the negative phase in (8.11) is not (guessing the visible nodes affect the hidden nodes and vice versa). Since $\mathrm{E}_h(\nabla E(\mathbf{v},\mathbf{h}|v))$ is tractable, the expectation w.r.t. $\mathbf{v}$ can be approximated by sampling. In contrastive divergence [66] (the version of approximate descent popular for RBMs), the sum of samples is often approximated with just one point estimate of $\tilde{\mathbf{v}}$:

$$-\frac{\partial\ln\left(p(\mathbf{v})\right)}{\partial\theta}\approx\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{v},\mathbf{h}\right)}{\partial\theta}|\mathbf{v}\right]-\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\tilde{\mathbf{v}},\mathbf{h}\right)}{\partial\theta}|\tilde{\mathbf{v}}\right] \tag{8.13}$$

**Approximate gradient descent in the RBM context:** The standard gradient descent update steps are summed over the $N$ training visible layer examples $\theta:=\theta-\alpha\sum_n\partial\ln p(\mathbf{v}_n)/\partial\theta$.

For efficiency, stochastic gradient descent is used; each update step involves exactly one of the $N$ training samples. In addition, it was found that it is not necessary to wait for the chain to converge [66], thus only a few steps (very often one) is enough. Gradient descent with these assumptions is called Contrastive divergence and it is summarized in algorithm 2.

In the style of the commonly used RBMs, we constrain our hidden variables to be binary in order to yield further simplified terms:

$$\mathrm{E}_{h_i}\left[\frac{\partial E\left(\mathbf{v},\mathbf{h}\right)}{\partial\{b_j,c_i,w_{ij}\}}|\mathbf{v}\right]=\left[-v_j,-\sigma\left(c_i+\mathbf{W}_i\mathbf{v}\right),-v_j.\sigma\left(c_i+\mathbf{W}_i\mathbf{v}\right)\right] \tag{8.14}$$

$$\text{Note: } p(h_i=1|\mathbf{v})=\sigma(c_i+\mathbf{W}_i.\mathbf{v})\,\text{sigmoid function} \tag{8.15}$$

The expected value of $h_i$ given $\mathbf{v}$: $\mathrm{E}_{h_i}[h_i|\mathbf{v}]=\int h_i p(h_i|\mathbf{v})dh_i=\sigma(c_i+\mathbf{W}_i\mathbf{v})$. Updates for the parameters can be computed jointly as:

$$\mathbf{b}:=\mathbf{b}+\alpha.(\mathbf{v}^0-\mathbf{v}^k)$$

$$\mathbf{c}:=\mathbf{c}+\alpha.(\sigma(\mathbf{c}+\mathsf{W}.\mathbf{v}^0)-\sigma(\mathbf{c}+\mathsf{W}.\mathbf{v}^k))$$

$$\mathsf{W}:=\mathsf{W}+\alpha.(\sigma(\mathbf{c}+\mathsf{W}.\mathbf{v}^0)\otimes\mathbf{v}^{0^{\top}}-\sigma(\mathbf{c}+\mathsf{W}.\mathbf{v}^k)\otimes\mathbf{v}^{k^{\top}})$$

Initialize $\theta$ randomly
**for** *each epoch* **do**
  Randomly shuffle training set
  **for** *each training sample* $\mathbf{v} := \mathbf{v}_n$ **do**
    Compute the approximate gradient for the RBM (eq. 8.13). To do this:

    - Set the visible layer of the RBM to the current training example $\mathbf{v}^0 = \mathbf{v}$, where $\mathbf{v}^k$ is the state of $\mathbf{v}$ after $k$ Gibbs steps.
    - Perform $k$ (ideally large, but $k = 1$ often sufficient) steps of Gibbs sampling as in (8.9), to get $\left\{\mathbf{v}^0 = \mathbf{v} = \mathbf{v}_n, \tilde{\mathbf{v}} = \mathbf{v}^k\right\}$. As shown in (8.14), the right hand side of (eq. 8.13) only needs $\left\{\mathbf{v}^0, \mathbf{v}^k\right\}$.

    Update: $\theta := \theta - \alpha \left(\frac{\partial - \ln p(\mathbf{v}_n)}{\partial \theta}\right)$.
  **end**
**end**

**Algorithm 2:** RBM Training

### 8.2.4 Completion using the RBM

After training, we have the learned parameters $\theta$. We **observe** a part of a test 3D shape $\mathbf{v}^g$ and we want to **estimate** the missing geometry $\mathbf{v}^e$. The initial visible layer: $\mathbf{v}^0 = [\mathbf{v}^g, \mathbf{v}^e]$ represents the partial shape. The RBM completion problem translates to estimating the *unobserved visible* nodes: $\hat{\mathbf{v}^e} = \text{argmax}_\theta \left(p(\mathbf{v}^e|\mathbf{v}^g, \theta)\right)$.

This case cannot be solved in closed form (remember, the nodes in the hidden layer are independent of each other only when the entire visible layer is observed). Therefore we adopt the approximate, generative, sampling based approach of [47] i.e. find samples from the RBM with the observed variables as close as possible to those of the test sample. We initialize $\mathbf{v}^e$ randomly and run $k$ Gibbs sampling steps[1] starting at $\mathbf{v}^0$, clamping the sampled values of $\mathbf{v}^{k^o} = \mathbf{v}^g$ in each iteration. The final solution always has the observed variables clamped to the observed values: $\hat{\mathbf{v}} = \left[\mathbf{v}^g, \mathbf{v}^{k^e}\right]$

## 8.3 Evaluation

We first introduce our dataset and experimental details for training/testing variations of our method and other standard techniques. We then finally evaluate

[1] We used $k = 10$. This works well in our experiments.

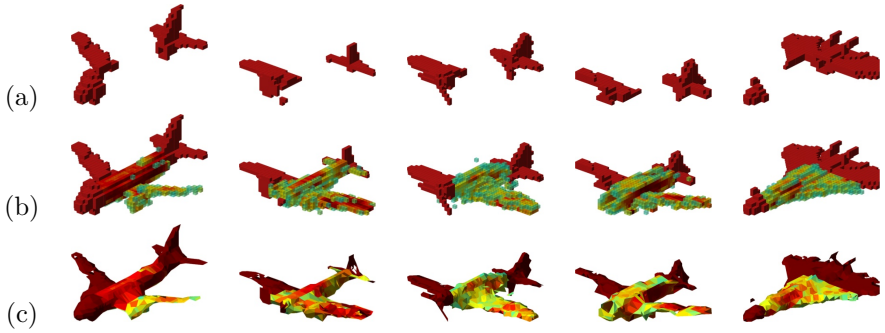



Figure 8.4: **Shape completion.** (a) Cropped shape used as an input, similar to fig. 8.1. $K = 3$ steps of Gibbs sampling are used to estimate the most probable missing data shown in (b) using an RBM (qualitatively, our RBM results look very similar). Estimated 3D shape is shown at (c). Color represents the probability of the reconstructed part, see colorbar in fig. 8.1.

them in comparison.

**Datasets:** Deep learning techniques including the RBM often have many parameters, needing a large amount of training data training examples [92, 47]. Therefore standard 3D shape benchmarks [166, 21] that have no more than one hundred (or less) shapes per class are not sufficient. Due to this, we evaluate the proposed method on the dataset of Huang *et al.* [70]. We used 1300 shapes per class (the car and airplane classes are used here). For each class, 800 shapes are used for training and the rest (300) for testing. The dataset provides shapes that are already in the same canonical frame of reference, thus avoiding the conflation of view estimation and shape completion problems. Each shape is voxelized at the resolution: $100 \times 100 \times 100 = 1M$ voxels. For efficiency of learning and prediction we remove the voxels that are never observed to have foreground/shape from the learning and estimation process, helping us reduce our nodes to $J \approx 40K$ on average. Each $J$ dimensional shape voxelization in vector form, defines the visible layer of variables $\mathbf{v}_n$. The visible layer is fully observed during training and partially during testing.

We can visualize results in two ways:

(i) Plotting translucent voxels, where its color and transparency corresponds to its probability (we show voxels $p(v_i = 1) > 0.3$) as in figures 8.1, 8.5, 8.7.

(ii) The voxelized result is meshed to view the result in surface form (fig. 8.1). For creating mesh structure from the set of voxels, we used the ball-

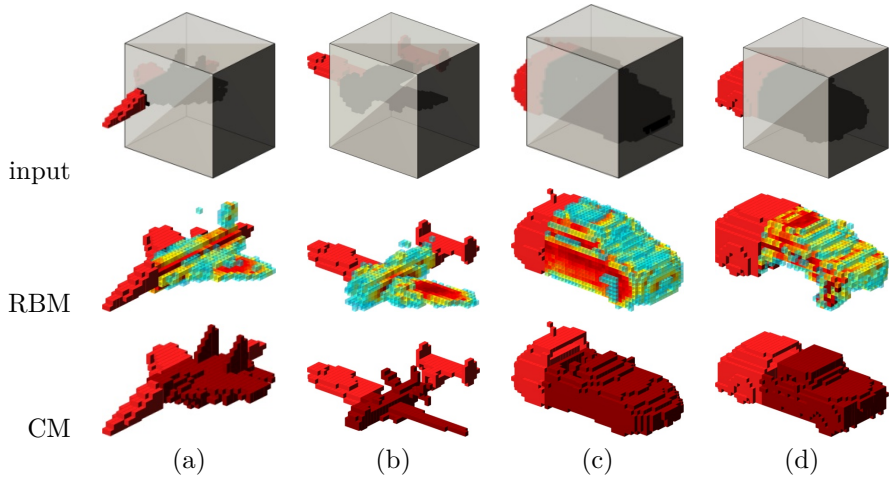



Figure 8.5: **Examples of shape completion.** Though CM performs well when it has similar instances in the training database, RBMs are better in completing shapes that were never seen before. The figure shows a few interesting cases of shape completion where CM outperforms RBM (a,d) and where RBM beats CM (b,c).

pivoting algorithm [15]. Note, superb mesh results are unlikely given the coarseness of the voxelization. With improved optimization in the future, we should be able to integrate more "continuous" Signed Distance representations in the entire process for improved results.

### 8.3.1 The competing techniques

The quality of the training and testing of any method is evaluated by:

(i) Qualitative visualization of generated samples: well-learnt RBMs should synthesize realistic shape instances.

(ii) Quantitative reconstruction error: The average absolute error between the ground truth and predicted shapes for synthesized hole regions.

We train and test three variations of our method: RBM, DBN and SBM (Shape Boltzmann machines [47]). In the absence of easily available benchmarks (as mentioned in §8.1) we evaluate variations of our method against three relatively simple but robust shape completion methods. The four schemes are stated below:

**Mean shape (MS):** As the naive competitor, we used the mean shape model (as in [47]), where each voxel is established independently. This can be expressed by an energy function $E(v, \theta) = \sum_i f_i(v_i; b_i)$, which we model by a RBM with one hidden node, $E(v; \theta) = \sum_i b_i v_i + \sum_i w_i v_i = \sum_i (2 b_i w_i) v_i$.

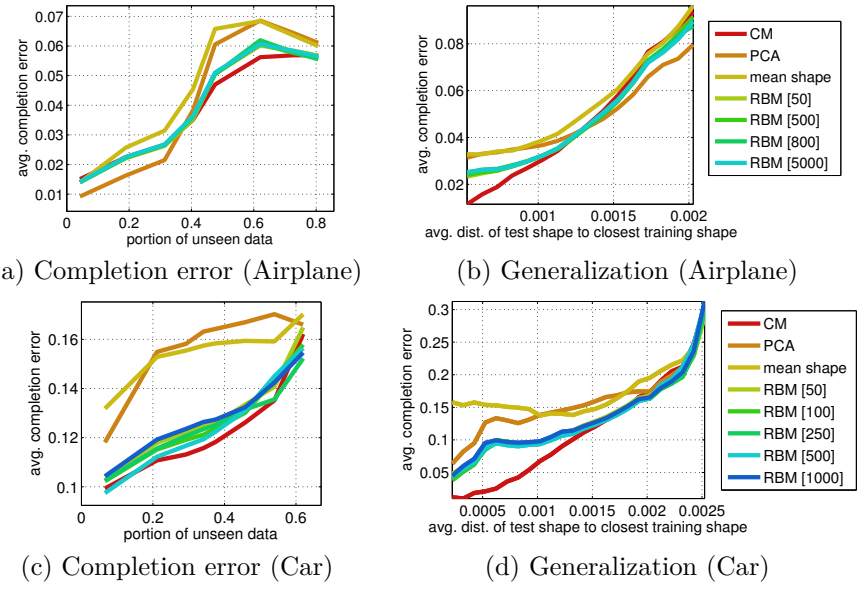
**Closest match (CM):** Motivated by [136], we also compared against the following non-parametric database driven method. For a test shape, we use the nearest neighbor training shape (with respect to similarity of observable voxels). The unobserved voxels of the test shape are filled by the nearest neighbor shape's counterpart. This method assumes a high similarity between the test shape and the database as it does not generalize. Obviously, it gets good results when the assumption holds, while the result weakens when the test shape differs to the database (fig. 8.5).

**PCA:** Principal components analysis allows us to learn a visible shape subspace such that every visible example can be approximated by $M$ eigen vectors, $\mathbf{v}_{\mathsf{pca}} = \sum_{k=1}^{M} \mathsf{U}_k^\top \mathbf{w}$, where $\mathsf{U}$ is a matrix of eigen vectors calculated from the training data and each 3D shape is represented by coefficients of the vector $\mathbf{w} = \mathsf{U}^\top \mathbf{v}^g$. Vector $\mathbf{v}^g$ represents the observed part of a test 3D shape.

**RBM:** Our RBM builds on [134] and it is trained on the dataset described above according to §8.2.3. A well-trained RBM can be used for subsequent completion as in §8.2.4.

### 8.3.2 Evaluating

We simultaneously increase the cropped area of test shapes (in steps of 20%) and use completion algorithms to predict the missing part, as shown in fig. 8.7. The performance corresponds to the absolute error between the ground truth and estimated voxels in the hole/cropped region. This allows us to relate the performance with the percentage of missing data as shown in fig. 8.6(a,c) for planes and cars. Each test shape is also associated with the distance to the most similar training shape. This allows plotting of the performance of the algorithm with respect to the specific similarities of test shapes to the training data. In this scenario, higher distances to the most similar training shape requires more generalization. Results are shown in fig. 8.6(b,d). As expected, CM gives very good results when test shapes are similar to the training data, while RBM performs better when generalization is needed. Surprisingly, PCA performs very well for the airplane class when generalization results, but only when a




(a) Completion error (Airplane) (b) Generalization (Airplane)

(c) Completion error (Car) (d) Generalization (Car)

Figure 8.6: **Shape completion error.** Plots (a,c) capture the average completion error with respect to cropped area (size of the cropped area). Plots (b,d) plot the same completion error but as a function of the L2 distance between the voxelized test shape and its closest training match. This allows us to evaluate the generalization ability of methods. For low distances, CM naturally performs well, but at high distances RBMs/PCA perform better. Across all plots PCA generalizes well (b,d), but performs poorly in (c,d).

small part of the shape is missing, see fig. 8.6(a,b). Fig. 8.7 visually presents results that corresponds to plots in fig. 8.6.

**Evaluating training** The RBM learning based schemes need to be trained effectively, and the right choice of the *fixed* parameters e.g. epochs, hidden units per layer, number of layers etc., is crucial. We summarize our findings for training:

(i) Number of visible variables: Figure 8.8(d) shows results for different dimensions of the visible layer. To be able to perform this experiment we voxelized shapes into varying resolution of the discretization grid. Note that it also affects the level of detail of the object and, of course the machine needs to learn it. Results are as expected, when the level of detail increase, the network with same parameters performs worse.

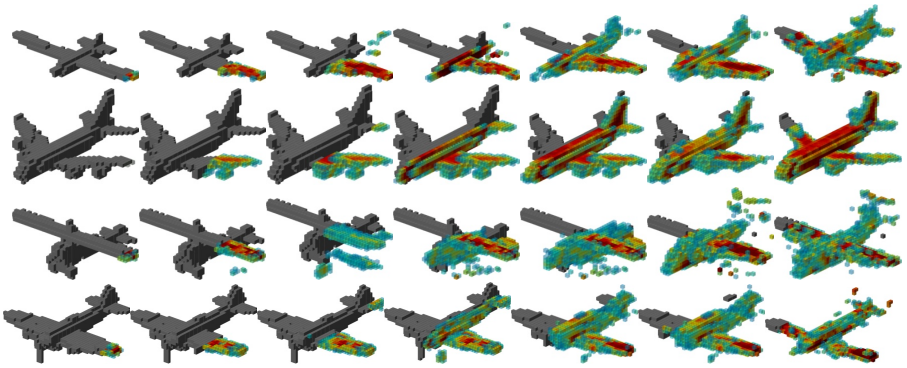


Figure 8.7: **Shape completion using RBM given varying missing regions.** The area of the hole is increased in steps of 20% progressively along the columns, the partial input is shown in gray. The proposed voxels are plotted, colored according to their probability as fig. 8.1. Interestingly, the hypothesized number of engines in the left hole of row 2 is seen to base itself on the number visible on the right. Expectantly, the red (high probability regions) are fewer when the hole is larger. Also, a large hole causes the hypothesized shape to be different from the ground truth, though it still looks plausible.

(ii) The number of hidden variables: The number of hidden variables are related to the modeling power of RBM and varies from one thousand nodes in the image completion example of [47] to millions for the image classification application of [92]. In 3D completion example in fig. 8.8(b), we found that more than 100 hidden nodes perform well.

(iii) Number of layers: Though increasing layers should increase the modeling power, learning becomes difficult without considerably more data. In our case, the difference between DBN and RBM performance differed by hardly 1%. Dropout like schemes (like SBM) are easier and faster to learn as it has significantly lower number of variables to learn. Unfortunately, we observed that average reconstruction error using SBM is close to MS in our experiments.

(iv) Number of epochs: We learn several RBM models with increasing number of epochs ([155]) and measure average reconstruction error on the same settings as above. Results are shown in fig. 8.8(a) for RBM with 1K hidden nodes, 200 hidden nodes and 100 hidden nodes. The difference between these RBMs is minimized when more epochs are performed and the error also decreases with a higher number of epochs. Surprisingly, when only a few epochs are used (tens) the method performs worst then

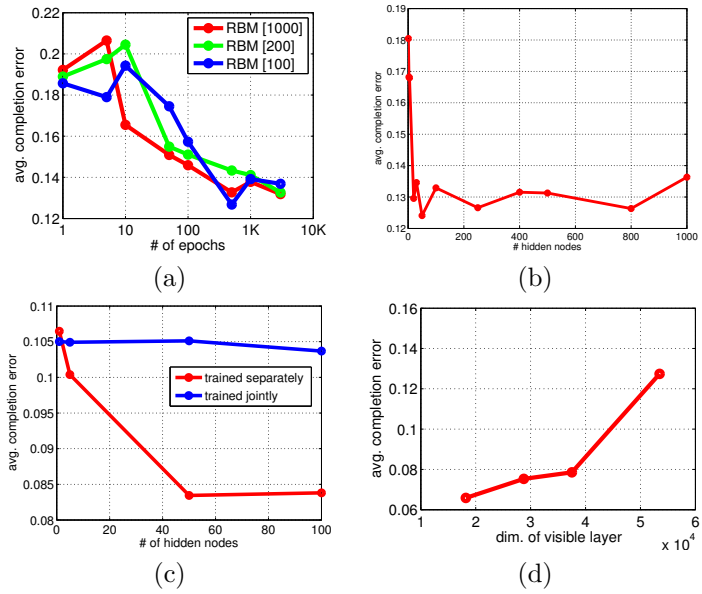



Figure 8.8: **The effect of RBM parameters on the completion performance.** (a) Relation of the average completion error to the number of epochs. As you see from the graph, spending more time on training improves the results. (b) Completion performance for different sizes of RBM. Note that extremely small RBMs perform badly, while after some limit, adding more hidden nodes does not help. (c) Performance of learning classes jointly and separately. Deep learning cannot learn classes at once, but different models for different classes are needed. (d) Relation of the completion performance to the dimension of the voxelization grid. This corresponds to the number of visible nodes. We left the all other parameters (such as the number of hidden nodes) same. Note that it is harder to complete properly when the visible space increases as there.

using each model only once. As a result, using more than 1000 epochs is necessary to get better results.

(v) Learning classes together: So far, we learn two separate models, one for the airplane class and a second for the car class. Models are applied for the class of shape they have been trained on. We explore whether the RBM can learn several classes together. For this experiment, we trained a RBM on the union of cars and planes (1.6K training shapes altogether). While it does not affect results for a few hidden nodes, we observed a decrease in performance when one RBM is learn for the union of classes for more than ten hidden nodes (fig. 8.8(c)).

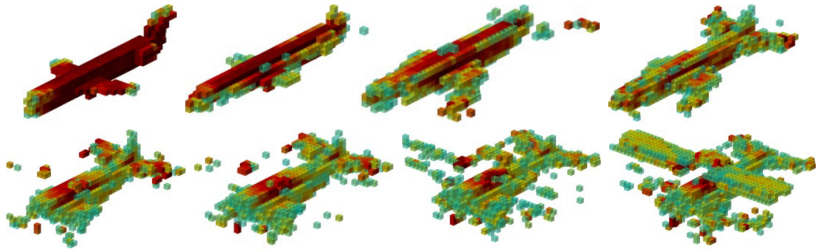


Figure 8.9: **What do hidden nodes represent?** We randomly turn on one hidden node after each other and we display activated visible nodes as voxels. For example, first figure shows the visible layer for one hidden node set to one (the hidden node was selected randomly), the second figure shows visible layer for two activated hidden nodes etc. Note the variability of shapes and how they deform from one sub-class of the plane to another.

### 8.3.3 Relations between hidden and visible nodes

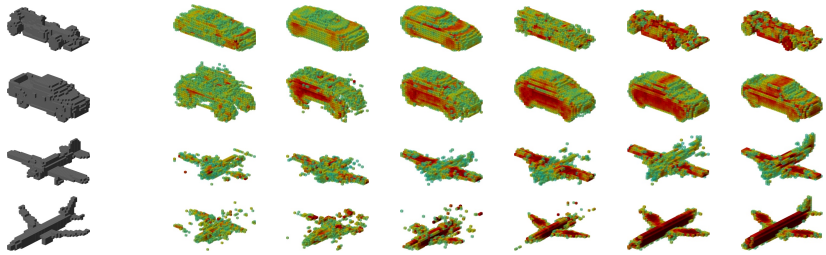
We show two simple examples of how several combinations of activating hidden nodes affects the visible layer and what happens when some hidden nodes are turned on or off.

In fig. 8.9 we illustrate an example where a few hidden nodes are randomly activated, and we iteratively activate more and more hidden nodes. The example shows that different visual modes go one after each other and that certain parts of the shape are added or discarded when different hidden nodes are on.

The learning quality of the RBM can be gauged by the quality of the samples it can generate. We attempt the following. Using a training example shape, the expected hidden layer is computed. Starting from this initial estimate of the hidden layer, a few Gibbs samples are obtained (without any constraints on the visible layer) only when nodes are randomly activated. The results are plotted for an increasing number of activated hidden nodes. As expected, the shape varies gradually between visual modes (fig. 8.10).

## 8.4 Improving scene's objects

When SfM or LIDAR are used to create a 3D model of the real word, they always suffers from missing parts, inaccuracies etc. Thus, an appropriate algorithm should to be used [136, 191]. Moreover, 3D data is usually in the form of a point cloud, so faces are missing [186] and, as expected, algorithms to estimate faces from the point cloud are not precise. In this context, it seems obvious to apply



(a) Visible layer (b) Increasing number of activated hidden nodes

Figure 8.10: **How does RBM represent 3D shape?** RBM hidden nodes are activated from the training shapes at (a). We activate only a subset of hidden nodes that correspond to (a). The most left figure of (b) corresponds to the one hidden node activated and the most right figure corresponds to the activation of all hidden nodes. The reconstructed shapes (b) evolve from the class general shape (left) to the object specific shape (right).

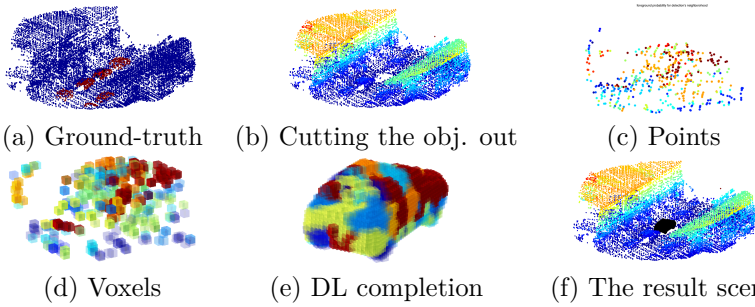
the previously introduced algorithm that learns relations between 3D points and use such models to model an improved noisy 3D reconstruction. Note that the generalization ability of the algorithm is necessary, as the object we want to improve was not seen before.

To improve such a noisy 3D reconstruction, we now explore the combination of the previous findings in object detection/segmentation where these findings have powerful 3D shapes completion. The **basic idea** is as follows: given a noisy point cloud as a 3D reconstruction of the scene, we run our joint object detection and segmentation algorithm from Chapter 7. Its result is a set of hypothesized object locations as well as the assignment of the scene's points to the specific objects or to the background. Then, segmented objects are automatically selected and voxelized, and we run the completion algorithm to estimate whether there are any missing parts and to determine which already occluding parts are incorrect. As a result of this method, the level of detail of the 3D object is corrected and enriched.

### 8.4.1 The method & results

The algorithm for gleaning shapes in the scene consists of putting already presented methods together with the following pipeline.

**Joint object detection & segmentation.** The goal of object detection is to obtain a location where the object occurs. For the task of shape completion,



(a) Ground-truth (b) Cutting the obj. out (c) Points

(d) Voxels (e) DL completion (f) The result scene

Figure 8.11: **Completion of objects in the scene.** Detection algorithm (ISM in this case) gives a position of the object in the scene (a). Part of the scene that is likely to be an object is separated (c,d) and used as the input for a completion algorithm (e). If object points have the probability of being foreground, we project these points into a voxel space and sample from this distribution (e). Using the completion algorithm we obtain an estimated model (f) that can be entered into a scene instead of the initial noisy points (g). In this example, we replace each estimated voxel with the point, so completed model is a point cloud as the scene.

the object needs to be segmented from the background as well. For this, we used the joint segmentation and detection presented in Chapter 7. Given a test scene and a training set, we obtain the location of an object and the approx. probability of the points belonging to the object (fig. 8.11(a)). As input for the next completion stage, we take a set of points around the estimated object location within the radius of the maximal object size (fig. 8.11(b,c)). Thus, each point, **i**, has its position and the approx. probability as to whether it belongs to the object class $Q_i(x_i = \ell)$.

**Completion.** The RBM for completion is applied on the cube of the voxel space. Thus, we need to transform 3D points into a voxel-space. Firstly, the set of points around the estimated object location is aligned using PCA and then normalized such that the maximal size of the object fits the dimension of the cube (fig. 8.11(d)). Once the data is in the cube, the voxel values correspond to $Q_i(x_i = \ell)$. We now perform Gibbs sampling in the same manner as in §8.2.4. The result shape is shown in fig. 8.11(e).

Deep Learning techniques require enormous amount of data for learning [67, 47]. We learn RBM on 1K models of cars in the Huang dataset [70] using the same settings as in §8.3. Results are shown in fig. 8.11.

## 8.5 Summary

Deep learning has provided us with the ability to learn complex features and relationships in data in an unsupervised way. Though similar neural network architectures existed before, optimization techniques that are both powerful, simple, and effective have made a huge difference. In this chapter, we were able to show the power of RBMs in the context of shape completion. Additionally, we attempted to analyze the data and study the effects of the fixed parameters in this process. The results correspond to our expectations. Being able to study the shape completion result in isolation from the problem of view estimation shows that, effective completion is possible even when a large part of the object is invisible. Rigid and deformable matching and correspondence estimation continues to be difficult, both in isolation (or jointly with the completion problem). There is no doubt that the Lego like results are a bit simplistic for visualization applications. Future work points to perhaps performing such learning entirely in a signed distance function space for improved results. Adaptive methods for shape representation and learning are also key. However, this first experiment in this space shows a clear possibility for integration of shape representation with related problems of recognition and reconstruction for a joint approach to vision.

# Chapter 9

# Conclusion & Future Work

While most popular methods for processing 3D data and learning from 3D databases used global information only, in this thesis, we have explored methods based on the local geometry of 3D shapes for classification, retrieval, and segmentation.

We have investigated the advantages of spatial constraints, especially considering the relative position of the object's features to the object centre. We have performed experiments on popular artificial hand-made benchmarks as well as on captured real-life data. A practical use of the method in 3D retrieval was shown for the 3D Coform [4] project. The project also opened a number of opportunities to interact with end users and to investigate questions that require their point of view.

The proposed method for 3D shape search and recognition is based on the extension of the image processing feature detector and descriptor (SURF [14]) into the third dimension. Taking the advantage of these robust local 3D SURF features, we learn their relative positions and appearances from the previously described applications.

In the last chapter, we have presented its application for 3D completion, but we have shown that a Deep Learning approach–that has just become state-of-the-art in computer vision–has huge promise in this particular application of 3D completion.

## 9.1 Future work

Methods for analyzing 3D data are popular mainly due to their extensive use in the gaming and movie industries. The recent rise of the availability of relatively large-scale datasets of 3D models, has placed methods for 3D data into the center of interest for machine learning and computer vision researches. In contrast to pure 2D data, 3D data is invariant to the material of the object, illumination, and texture. Thus, analyzing data in 3D has the potential to improve results significantly, as extreme differences in the aforementioned properties are the current bottle neck of 2D methods [49, 92]. However, although 3D information is just becoming useful to improve Structure-from-Motion [11], image segmentation [174] or object detection [139], these approaches still heavily rely on 2D features. Some even show that 2D information is much more important than 3D descriptors [184]. In contrast, recent works show that pure 3D geometry analysis or pure 3D pre-segmentation–problems that this thesis focuses on–really introduce advantages over 2D based methods. For example, recognizing objects in 3D space outperforms 2D state-of-the-art [174], or processing data in reconstructed 3D point clouds introduces a dramatic speed-up (20x) with competitive results [114] compared to the methods that focus on 2D data only.

We have also presented initial experiments with Deep Learning, where no need of feature pre-calculation (which is often the bottle neck of the method) is required, but more principled connections are learnt instead. We hope that this opens doors for more applications and further research as these directions already defined state-of-the-art for image processing [92, 47].

# Bibliography


[1] Activities for kids and teachers: All about butterflies. `http://kiddyhouse.com/butterflies/`. [Online; accessed July-2014]. pages 8

[2] FVS: A content-based retrieval library. `http://sourceforge.net/projects/libfvs/`. [Online; accessed July-2014]. pages 14, 35

[3] Encyclopaedia Britannica. 1952. pages 7, 8

[4] 3D-COFORM: Project final report. 2012. pages 4, 5, 121

[5] 3dRender.com. Lighting challenges. `http://www.3drender.com/challenges/`. [Online; accessed May-2011]. pages 82

[6] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics*, 24:408–416, July 2005. pages 102, 103

[7] R. Arandjelović and A. Zisserman. Efficient image retrieval for 3D structures. In *Proc. BMVC.*, 2010. pages 55

[8] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *ACM Symposium on Discrete Algorithms*, 45, 1998. pages 95

[9] D. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981. pages 8

[10] H. Y. Bang and T. Chen. Feature space warping: An approach to relevance feedback. In *Intl. Conf. Image Proc.*, volume 1, 2002. pages 54

[11] S. Y. Bao and S. Savarese. Semantic structure from motion. In *Proc. CVPR*, 2011. pages 104, 122

[12] Y. Bao, M. Chandraker, Y. Lin, and S. Savarese. Dense object reconstruction using semantic priors. In *Proc. CVPR*, 2013. pages 1

[13] P. Bariya and K. Nishino. Scale-hierarchical 3D object recognition in cluttered scenes. In *Proc. CVPR*, 2010. pages 74

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 2008. pages 3, 5, 14, 21, 24, 25, 26, 121

[15] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 1999. pages 111

[16] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Comput. Surv.*, 1985. pages 8, 11

[17] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. pages 16, 64

[18] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proc. CVPR*, 2008. pages 16, 80, 97

[19] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE PAMI*, 26(9):1124–1137, September 2004. pages 74, 75, 77, 81, 89, 97

[20] A. Bronstein, M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 2008. pages 18, 41, 62, 82

[21] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape Google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph.*, 2011. pages 12, 16, 52, 54, 55, 56, 62, 110

[22] R. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence Journal*, 17(1-3):285–348, 1981. pages 1, 8

[23] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, 2008. pages 23, 75

[24] A. Brunton, S. Wuhrer, C. Shu, P. Bose, and E. D. Demaine. Filling holes in triangular meshes using digital images by curve unfolding. *International Journal of Shape Modeling*, 16, 2010. pages 103

[25] N. D. F. Campbell, K. Subr, and J. Kautz. Fully-connected CRFs with non-parametric pairwise potentials. In *CVPR*, 2013. pages 89

[26] M. Casale and E. Stanton. An overview of analytic solid modeling. *IEEE Computer Graphics and Applications*, 1985. pages 11

[27] I. Chakravarty and H. Freeman. Characteristic views as a basis for three-dimensional object recognition. *Proceedings of SPIE*, 1982. pages 12

[28] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. pages 37, 63

[29] S. Chaudhuri, E. Kalogerakis, L. J. Guibas, and V. Koltun. Probabilistic reasoning for assembly-based 3D modeling. *ACM Trans. Graph.*, 30(4), 2011. pages 4, 10

[30] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. In *SIGGRAPH*, 2013. pages 88

[31] W. Choi, Y. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3D geometric phrases. In *Proc. CVPR*, 2013. pages 10

[32] O. Chum, A. Mikulik, M. Perdoch, and J. Matas. Total recall II: Query expansion revisited. In *Proc. CVPR*, 2011. pages 54, 55, 58

[33] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *Proc. CVPR*, 2009. pages 56

[34] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proc. CIVR*, 2007. pages 14

[35] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*, 2007. pages 15

[36] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *ICDAR*. IEEE, 2011. pages 16

[37] J. Cui, F. Wen, and X. Tang. Real time Google and live image search re-ranking. In *ACM Multimedia*, 2008. pages 55

[38] C. M. Cyr and B. B. Kimia. A similarity-based aspect-graph approach to 3D object recognition. *IJCV*, 57, 2004. pages 12, 14

[39] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *3D Data Processing Visualization and Transmission*, 2002. pages 102, 103

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. CVPR*, 2009. pages 2

[41] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *Proc. ACM SIGGRAPH*, 2003. pages 103

[42] M.-P. Dubuisson and A. K. Jain. A modified Hausdorff distance for object matching. In *Proc. ICPR*, 1994. pages 56

[43] H. Dutagaci, A. Godil, A. Axenopoulos, P. Daras, T. Furuya, and R. R. Ohbuchi. SHREC 2009 - Shape retrieval contest of partial 3D models. In *Eurographics Workshop on 3D Object Retrieval*, 2009. pages 15, 18, 34, 35, 62

[44] M. Efron. Query expansion and dimensionality reduction: Notions of optimality in rocchio relevance feedback and latent semantic indexing. *Inf. Process. Manage.*, 2008. pages 54

[45] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa. Sketch-based shape retrieval. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 2012. pages 52

[46] M. Elad, A. Tal, and S. Ar. Content based retrieval of VRML objects: An iterative and interactive approach. In *Proceedings of the sixth Eurographics workshop on Multimedia 2001*. Springer-Verlag, 2002. pages 54

[47] S. M. A. Eslami, N. Heess, C. K. I. Williams, and J. Winn. The shape Boltzmann machine: A strong model of object shape. In *IJCV*, 2013. pages 102, 103, 104, 107, 109, 110, 111, 112, 114, 118, 122

[48] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005. pages 89

[49] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE PAMI*, 2010. pages 1, 16, 38, 39, 56, 122

[50] M. Fisher and P. Hanrahan. Context-based search for 3D models. *SIGGRAPH Asia*, 2010. pages 52

[51] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2004. pages 52, 102

[52] T. Funkhouser and P. Shilane. Partial matching of 3D shapes with priority-driven search. In *Eurographics symposium on Geometry processing*, 2006. pages 42

[53] T. Furuya and R. Ohbuchi. Dense sampling and fast encoding for 3D model retrieval using bag-of-visual features. In *Proc. CIVR*, 2009. pages 12, 13, 14

[54] S. C. Gad. *Drug discovery handbook*. Wiley-Interscience, 2005. pages 16

[55] J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. In *Proc. CVPR*, 2009. pages 95

[56] P. Gehler and O. Chapelle. Deterministic annealing for multiple-instance learning. In *Artificial Intelligence and Statistics (AIStats)*, 2007. pages 53

[57] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proc. ICCV*, 2009. pages 35, 36

[58] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proceedings of the third Eurographics symposium on Geometry processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association. pages 22, 42

[59] D. Gennery. Object detection and measurement using stereo vision. In *International Joint Conference on Artificial Intelligence*, 1979. pages 9

[60] Gizmondo. What facebook deals with everyday. `http://gizmodo.com/5937143/what-facebook-deals-with-everyday-27-billion-likes-300-million-photos-uploaded-and-500-terabytes-of-data`. [Online; accessed February-2015]. pages 2

[61] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. In *Proc. ICCV*, 2009. pages xix, 1, 4, 10, 17, 18, 23, 75, 83, 88, 89, 95, 97, 98

[62] Google Inc. Google 3D Warehouse. `http://sketchup.google.com/3dwarehouse/`. pages 1, 19, 52, 66, 67

[63] Google Inc. Google Earth. `https://www.google.com/earth/`. [Online; accessed August-2014]. pages 2

[64] Google Inc. Google Street View. `http://www.google.com/maps/about/behind-the-scenes/streetview/`. [Online; accessed August-2014]. pages 2

[65] X. He, R. S. Zemel, and M. A. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. *Proc. CVPR*, 2004. pages 89

[66] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14, 2002. pages 105, 107, 108

[67] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012. pages 103, 118

[68] S. C. H. Hoi and R. Jin. Active multiple kernel learning for interactive 3D object retrieval systems. *ACM Trans. Interact. Intell. Syst.*, 2011. pages 54

[69] R. Horaud and R. Bolles. 3dpo's strategy for matching three-dimensional objects in range data. In *In Proceedings of the International Conference on Robotics*, 1984. pages 9

[70] Q.-X. Huang, H. Su, and L. Guibas. Fine-grained semi-supervised labeling of large shape collections. *ACM Trans. Graph.*, 2013. pages 15, 18, 101, 110, 118

[71] X. Huang, H. Fu, O. K.-C. Au, and C.-L. Tai. Optimal boundaries for Poisson mesh merging. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, New York, NY, USA, 2007. ACM. pages 102, 103

[72] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011. pages 1, 2, 10, 73, 102

[73] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 2010. to appear. pages 12, 28, 29, 55

[74] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE PAMI*, 21(5):433–449, 1999. pages 12, 13, 21, 35, 42, 53, 56, 74, 95

[75] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model of component-based shape synthesis. *ACM Transactions on Graphics*, 2012. pages 102

[76] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(3), 2010. pages 4, 12, 17, 75, 89

[77] N. Kholgade, T. Simon, A. A. Efros, and Y. Sheikh. 3D object manipulation in a single photograph using stock 3D models. In *Proc. ACM SIGGRAPH*, 2014. pages 2

[78] B. Kim, S. Xu, and S. Savarese. Accurate localization of 3D objects from RGB-D data using segmentation hypotheses. In *Proc. CVPR*, 2013. pages 102, 103

[79] V. G. Kim, W. Li, N. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3D models using fuzzy correspondences. *Transactions on Graphics (Proc. of SIGGRAPH 2012)*, 2012. pages 55, 56

[80] V. G. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. *ACM Transactions on Graphics*, 2011. pages 65

[81] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics*, 2012. pages 88, 89

[82] J. Knopp, M. Prasad, and L. Van Gool. Orientation invariant 3D object classification using Hough transform based methods. In *Proceedings of the ACM workshop on 3D object retrieval*, 3DOR '10. ACM, 2010. pages 15, 56

[83] J. Knopp, M. Prasad, and L. Van Gool. Scene cut: Class-specific object detection and segmentation in 3D scenes. In *3DIMPVT*, 2011. pages 6, 73, 88, 89, 90, 95

[84] J. Knopp, M. Prasad, and L. Van Gool. Automatic shape expansion with verification to improve 3D retrieval, classification and matching. In *Eurographics - WS on 3D Object Retrieval*, 2013. pages 5, 16, 52

[85] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. J. V. Gool. Hough Transform and 3D SURF for robust three dimensional classification. In *Proc. ECCV*, 2010. pages 5, 8, 12, 16, 18, 21, 43, 52, 54, 56, 57, 58, 61, 62, 74, 75, 76, 78, 82

[86] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *Proc. ECCV*, 2010. pages 5, 41

[87] L. Kobbelt, P. Schraider, M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, 2003. pages 12, 13, 14, 22, 35, 42

[88] J. J. Koenderink and A. J. van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24:51–59, 1976. pages 12

[89] P. Kohli, L. Ladický, and P. Torr. Robust higher order potentials for enforcing label consistency. *Int. J. Comput. Vision*, 82(3), 2009. pages 89

[90] N. Komodakis and G. Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Transactions on Image Processing*, 16(11), 2007. pages 103

[91] P. Krahenbuhl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011. pages 17, 89

[92] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *NIPS*. 2012. pages 1, 16, 103, 107, 110, 114, 122

[93] D. T. Kuan and R. J. Drazovich. Model-based interpretation of range imagery. In *AAAI*. AAAI Press, 1983. pages xvii, 8, 9

[94] M. P. Kumar, P. Torr, and A. Zisserman. Objcut: Efficient segmentation using top-down and bottom-up cues. *IEEE PAMI*, 32, 2009. pages 17, 18

[95] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proc. CVPR*, volume 1, pages 18–25, 2005. pages 74

[96] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Objcut: Efficient segmentation using top-down and bottom-up cues. In *IEEE PAMI*, volume 32, 2010. pages 89

[97] S. Kumar and M. Hebert. A hierarchical field framework for unified context-based classification. In *Proc. ICCV*, 2005. pages 89

[98] L. Ladický, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical CRFs for object class image segmentation. In *Proc. ICCV*, 2009. pages 89

[99] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001. pages 89

[100] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *Proc. ICML*, 2012. pages 103

[101] A. Lehmann, B. Leibe, and L. V. Gool. Feature-centric efficient subwindow search. In *Proc. ICCV*, 2009. pages 28, 38, 41, 43

[102] A. Lehmann, B. Leibe, and L. Van Gool. PrISM: Principled Implicit Shape Model. In *Proc. BMVC.*, 2009. pages 80, 95

[103] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004. pages 90, 95

[104] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77, 2008. pages 8, 12, 16, 22, 28, 29, 31, 38, 41, 43, 48, 61, 74, 75, 78, 85

[105] F. F. Leymarie and B. B. Kimia. The shock scaffold for representing 3D shape. In *Workshop on Visual Form (IWVF4)*, 2001. pages 22

[106] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-Independent Object Class Detection using 3D Feature Maps. In *Proc. CVPR*, 2008. pages 1

[107] K. Lou, S. Jayanti, N. Iyer, Y. Kalyanaraman, S. Prabhakar, and K. Ramani. A reconfigurable 3D engineering shape search system: Part ii — Database indexing, retrieval, and clustering. *ASME Conference Proceedings*, 2003, 2003. pages 54

[108] D. G. Lowe. The viewpoint consistency constraint. *IJCV*, 1(1):57–72, 1987. pages 1, 3, 8

[109] B. Luo, X. Wang, and X. Tang. A world wide web based image search engine using text and image content features. In *Proc. of IS&T/SPIE Electronic Imaging*, 2003. pages 55

[110] S. Maji and J. Malik. Object detection using a max-margin Hough transform. In *Proc. CVPR*, 2009. pages 30, 43

[111] T. Malisiewicz. *Exemplar-based Representations for Object Detection, Association and Beyond*. PhD thesis, Carnegie Mellon University, 2011. pages 8

[112] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. pages 14

[113] D. Marr. *Vision*. MIT Press, 1982. pages 8

[114] A. Martinovic*, J. Knopp*, H. Riemenschneider, and L. Van Gool. 3D all the way: Semantic segmentaion of urban scenes from start to end in 3D. In *Proc. CVPR*, 2015. pages 122

[115] D. J. Meagher. Geometric modeling using octree encoding. In *Comput. Graphics Image Processing*, 1981. pages 11

[116] D. L. Medin and M. M. Schaffer. Context theory of classification learning. In *Psychilogical Review*, 1978. pages 8

[117] A. S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE PAMI*, 28, 2006. pages 22, 44

[118] A. Mikulik, M. Perdoch, O. Chum, and J. Matas. Learning a fine vocabulary. In *Proc. ECCV*, 2010. pages 54

[119] P. Min, M. Kazhdan, and T. Funkhouser. A comparison of text and shape matching for retrieval of online 3D models. In *European Conference on Digital Libraries*, 2004. pages 55

[120] T. P. Minka and R. W. Picard. Interactive learning with a "society of models". In *Proc. CVPR*, 1996. pages 54

[121] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, 2009. pages 48, 58

[122] D. Munoz, N. Vandapel, and M. Hebert. Directional associative Markov network for 3-D point cloud classification. In *International Symposium on 3-D Data Processing, Visualization, and Transmission (3DPVT)*, 2008. pages 75

[123] K. Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics*, 2013. pages 11

[124] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2012)*, 2012. pages 88, 89

[125] Neptec. *Wright State 100*. `http://www.daytaohio.com/Wright_State100.php`, 2009. pages 97

[126] R. Nevatia and T. Binford. Description and recognition of complex curved objects. *Artificial Intelligence Journal*, 8, 1977. pages 1, 8

[127] J. Novatnack and K. Nishino. Scale-dependent/invariant local 3D shape descriptors for fully automatic registration of multiple sets of range images. In *Proc. ECCV*, 2008. pages 22

[128] J. O'Rourke and N. Badle. Decomposition of three-dimensional objects into spheres. *IEEE PAMI*, 1979. pages 12

[129] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobki. Shape distributions. *ACM Transactions on Graphics*, pages 807–832, 2002. pages 4, 12, 13, 14, 22

[130] M. Oshima and Y. Shirai. Object recognition using three-dimensional information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1983. pages 9

[131] M. Ovsjanikov, A. M. Bronstein, M. M. Bronstein, and L. J. Guibas. Shape Google: A computer vision approach for invariant shape retrieval. In *Workshop on NORDIA*, 2009. pages 4, 22, 32, 42, 43, 52, 74

[132] M. Ovsjanikov, W. Li, L. Guibas, and N. J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM Transactions on Graphics*, 2011. pages 52, 53

[133] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas. One point isometric matching with the heat kernel. In *Eurographics Symposium on Geometry Processing (SGP)*, 2010. pages 55, 56, 65

[134] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data. *Master thesis*, 2012. pages 112

[135] I. P. Papadakis, T. Trafalis, T. Theoharis, and S. Perantonis. Relevance feedback in content-based 3D object retrieval a comparative study. In *Computer-Aided Design & Applications*, 2008. pages 54

[136] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. Guibas. Example-based 3D scan completion. In *Symposium on Geometry Processing*, pages 23–32, 2005. pages 102, 103, 112, 116

[137] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics*, 27(3), 2008. pages 102, 103

[138] A. P. Pentland. Perceptual organisation and the representation of natural form. *Artificial Intelligence*, 1986. pages 1, 8

[139] B. Pepik, P. Gehler, M. Stark, and B. Schiele. $3D^2PM$ - 3D deformable part models. In *Proc. ECCV*, 2012. pages 1, 122

[140] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D geometry to deformable part models. In *Proc. CVPR*, 2012. pages 1

[141] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007. pages 1, 12, 14, 15, 16, 55, 56

[142] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*, 2008. pages 29, 41

[143] J. Philbin, J. Sivic, and A. Zisserman. Geometric lda: A generative model for particular object discovery. In *Proc. BMVC.*, 2008. pages 54, 60, 62

[144] J. Philbin, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *Proc. ECCV*, 2010. pages 15, 61, 62

[145] R. Pieraccini. *The Voice in the Machine. Building Computers That Understand Speech.* MIT Press, 2012. pages 16

[146] H. Pottmann, J. Wallner, Q.-X. Huang, and Y.-L. Yang. Integral invariants for robust geometry processing. *Comput. Aided Geom. Des.*, 26(1):37–60, 2009. pages 22

[147] M. Prasad, J. Knopp, and L. Van Gool. Class-specific 3D localization using constellations of object Parts. In *Proc. BMVC.*, 2011. pages 65

[148] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. J. V. Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *Proc. CVPR*, 2011. pages 60, 62

[149] H. Riemenschneider, S. Sternig, M. Donoser, P. M. Roth, and H. Bischof. Hough regions for joining instance localization and segmentation. In *Proc. ECCV*, 2012. pages 18

[150] J. Rocchio. Relevance feedback in information retrieval. In *The SMART retrieval system - Experimetns in Automatic Document Processinig*, 1971. pages 58

[151] W. D. Ross. Plato's theory of ideas. 1951. pages 7, 8

[152] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proc. CVPR*, 2005. pages 103

[153] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *Proc. ACM SIGGRAPH*, 23(3), 2004. pages 74

[154] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. pages 5

[155] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *AISTATS*, 2009. pages 114

[156] S. Salti, F. Tombari, and L. D. Stefano. On the use of implicit shape models for recognition of object categories in 3D data. In *Proc. Asian Conf. on Computer Vision*, 2010. pages 76, 78

[157] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 1988. pages 35

[158] M. Salzmann, R. Urtasun, and P. Fua. Local deformation models for monocular 3D shape recovery. In *Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, 2008. pages 102, 103, 104

[159] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *Proc. ICCV*, 2011. pages 1

[160] D. Saupe and D. V. Vranic. 3D model retrieval with spherical harmonics and moments. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, London, UK, 2001. Springer-Verlag. pages 22, 74

[161] S. Shafer and T. Kanade. The theory of straight homogeneous generalized cylinders and a taxonomy of generalized cylinders. In *Proceedings of the 1983 DARPA Image Understanding Workshop*, 1983. pages 12

[162] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 878–887, New York, NY, USA, 2004. ACM. pages 102, 103

[163] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM TOG (Proc. SIGGRAPH Asia)*, 31(6), 2012. pages 89

[164] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. CVPR*, pages 731–743, 1997. pages 17, 74

[165] P. Shilane and T. Funkhouser. Selecting distinctive 3D shape descriptors for similarity retrieval. In *Shape Modeling International*, 2006. pages 52

[166] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, 2004. pages 15, 18, 62, 110

[167] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *Proc. ICCV*, 2005. pages 74

[168] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. ECCV*, pages 1–15, 2006. pages 89

[169] H. Y. Shum and R. Szeliski. Stereo reconstruction from multiperspective panoramas. In *Proc. ICCV*, 1999. pages 10

[170] I. Sipiran and B. Bustos. A robust 3D interest points detector based on Harris operator. In *Proceedings of the 3rd Eurographics Conference on 3D Object Retrieval*, 2010. pages 12, 54

[171] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003. pages xviii, 1, 12, 14, 15, 16, 22, 28, 34, 37, 43, 48, 53, 55, 80

[172] P. Smolensky. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory. MIT Press, 1986. pages 104

[173] N. Snavely, S. Seitz, and R. Szeliski. Photo Tourism: Exploring photo collections in 3D. In *Proc. ACM SIGGRAPH*, 2006. pages 10, 52, 83

[174] S. Song and J. Xiao. Sliding shapes for 3D object detection in depth images. In *Proc. ECCV*, 2014. pages 1, 122

[175] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, 2004. pages 103

[176] B. I. Soroka and R. K. Bajcsy. A program for describing complex three-dimensional objects using generalized cylinders as primitives. In *Pattern Recognition and Image Processing Conference*, 1978. pages 12

[177] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *SGP*, 2009. pages 12, 13, 15, 22, 35, 42, 54, 56, 74, 82

[178] M. Sunkel, S. Jansen, M. Wand, and H.-P. Seidel. A correlated parts model for object detection in large 3D scans. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, 2013. pages 88, 89

[179] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools Appl.*, 2008. pages 12, 15

[180] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the DARPA Grand challenge. *Journal of Field Robotics*, 2006. pages 10

[181] R. Toldo, U. Castellani, and A. Fusiello. A bag of words approach for 3D object categorization. In *MIRAGE*, 2009. pages xxi, 12, 16, 23, 32, 35, 36, 37, 38, 42, 43, 46, 52, 53, 54, 64, 74, 76

[182] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. CVPR*, pages 762–769, 2004. pages 92, 95, 97

[183] J. K. Udupa and I. S. N. Murthy. New concepts for three-dimensional shape analysis. *IEEE Trans. Computers*, 1977. pages 12

[184] J. P. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. Torr. Mesh based semantic modelling for indoor and outdoor scenes. In *Proc. CVPR*, June 2013. pages 122

[185] A. Velizhev, R. Shapovalov, and K. Schindler. An implicit shape model for object detection in 3D point clouds. *22nd ISPRS Congress, Melbourne, Australia*, 2012. pages 88, 89, 97, 98

[186] M. Vergauwen and L. V. Gool. Web-based 3D reconstruction service. *Mach. Vision Appl.*, 17(6), 2006. pages 1, 10, 18, 38, 39, 52, 68, 83, 84, 116

[187] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, pages 511–518, 2001. pages 74

[188] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proc. ECCV*, pages 650–663, 2008. pages 24, 25, 26, 44, 46, 74, 82

[189] O. Woodford, M.-T. Pham, A. Maki, F. Perbet, and B. Stenger. Demisting the Hough transform for 3D shape recognition and registration. In *Proc. BMVC.*, 2011. pages 42

[190] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Proc. ICCV*, volume 1, 2005. pages 63

[191] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, 2013. pages 116

# Publications

- 3D all the way: Semantic segmentation of urban scenes from start to end in 3D.
  A. Martinović*, J. Knopp*, H. Riemenschneider, L. Van Gool.
  *In Proceedings of the Computer Vision and Pattern Recognition, 2015*

- Automatic shape expansion with verification to improve 3D retrieval, classification and matching.
  J. Knopp, M. Prasad, L. Van Gool.
  *In Eurographics workshop on 3D Object Retrieval, 2013*

- Class-specific 3D localization using constellations of object parts.
  M. Prasad, J. Knopp, L. Van Gool.
  *In BMVC, 2011* (oral)

- Scene Cut: Class-specific object detection and segmentation in 3D scenes.
  J. Knopp, M. Prasad, L. Van Gool.
  *In 3DIMPVT, 2011*

- Orientation invariant 3D object classification using Hough Transform based methods.
  J. Knopp, M. Prasad, L. Van Gool.
  *In ACM Multimedia workshop on 3D Object Retrieval, 2010*

- Hough Transform and 3D SURF for robust three dimensional classification.
  J. Knopp, M. Prasad, G. Willems, R. Timofte, L. Van Gool.
  *In Proceedings of the European Conference on Computer Vision, 2010*

- Avoiding confusing features in place recognition.
  J. Knopp, J. Sivic, T. Pajdla.
  *In Proceedings of the European Conference on Computer Vision, 2010*

* Indicates equal contribution.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF OF ELETRICAL ENGINEERING
PSI-VISICS
Kasteelpark Arenberg 10 box 2441
B-3001 Heverlee